CP/M 3

Ein Arbeitsbuch

Band 2

von Raoul O. Koerber

Inhaltsverzeichnis

Teil IV die hardware-Anpassung an CP/M 3

Kapitel 1 Was bedeutet schon Anpassung ...

Der größte Vorteil von CP/M ist sicherlich seine Flexibilität in der Anpassung an vorhandene Hardware. Aber genau darin liegt auch ein evtl. entscheidender Nachteil.

Ist es auf der einen Seite möglich, so gut wie jedes Terminal, jedes Video-Display oder jede Grafikkarte mit entsprechenden Treibern anzuschließen (genau so wie jede Serienschnittstelle oder jeden Drucker) so können auf der anderen Seite eben diese Geräte auch in ihren Möglichkeiten völlig verschieden sein.

Ein CP/M-Programm, das als Konsolenausgang z.B. ein Terminal mit Cursor-Adressierung verlangt, kann kaum in einem System reibungslos funktionieren, das mit einer einfachen Videokarte ausgerüstet ist, die eben keine Cursor-Adressierung hat. Ebenso nützt ein wunderhübsches Grafik-Programm einem Anwender ohne der passenden Hardware gar nichts.

Hier ist vor allem der Grund zu suchen, daß es unter CP/M so gut wie keine Bildschirmspiele gibt, wie diese doch auf jedem Billigstcomputer - auch noch in Farbe - ablaufen. Sicherlich sind derartige Programme möglich, eine kompatible Hardware vorausgesetzt, aber genau diese Art von Programmen ist und soll auch nicht die Domäne von CP/M sein.

Unter CP/M gibt es ein Vielzahl guter Programme in allen Preislagen, die es dem Benutzer möglich machen, alle Computeraufgaben zu lösen zu deren Zweck eine CP/M-Maschine erstanden wurde. Dieser 'Zweck' liegt meist in Textverarbeitungs- und Buchhaltungsaufgaben, aber auch in der Programmierung von Steuerungen der verschiedensten Art.

Bevor eine CP/M-Maschine funktioniert, sind aber die unterschiedlichsten Hardware-Voraussetzungen an das CP/M anzupassen und ein Hilfsprogramm bereitzustellen, das den Kaltstart (also das Booten einer CP/M-Systemdiskette) ermöglicht.

Es ist dabei zu berücksichtigen, welche Konsolenschnittstelle zur Verfügung steht, welche Schnittstelle zu den Diskettenlaufwerke und um welche Laufwerkstypen es sich dabei handelt. Diese Hardware-Anpassung wird in einem Programmteil des CP/M mit dem Namen BIOS (Basic Input Output System - Grundsystem für Ein- und Ausgabe) vorgenommen.

Alle notwendigen Anpassungen sind in den nachfolgenden Kapiteln erläutert und an Beispielen verdeutlicht.

Da die Hardware, wie bereits angedeutet, in einer CP/M-Umgebung sehr unterschiedlich ist, mußte eine bestimmte Konfiguration als Beispiel herangezogen werden, um auch eine wirklich funktionierende Software zeigen zu können, denn sonst wären nur 'Spielbeispiele' entstanden.

Ausgewählt wurde zu diesem Zweck der NDR-Klein-Computer mit einer Z80-CPU, einem Grafikinterface (oder einer Serienschnittstelle zu einem Terminal) und dem Kontrollerbaustein FD 1793 zur Diskettenbearbeitung.

Die Auswahl fiel deshalb auf das genannte Gerät, weil alle Unter-

lagen, wie Schaltpläne und Beschreibungen zu erhalten sind. Auch können alle benötigten Baugruppen in Form von Platinen, Bausätzen oder Fertiggeräten preisgünstig bezogen werden.

Alle benötigten Baugruppen sind typisch für einen CP/M-Computer, so daß die besprochene Software relativ einfach auf andere Computer übernommen bzw angepaßt werden kann.

Alle Assemblerlistings wurden mit dem Macro-Assembler Z80ASM erstellt, der weitgehend kompatibel zum Macro-Assembler M80 von Microsoft ist. Der Z80ASM hat jedoch den Vorteil erheblich schneller, flexibler und preisgünstiger zu sein.

Die Programme wie auch alle anderen Beispiele wurden mit Z80-Mnemonics geschrieben.

Kapitel 2 Nach dem Einschalten ...

Jeder Computer benötigt sofort nach dem Einschalten Anweisungen was er nun tun soll. Diese Anweisungen erhält er normalerweise von einem Monitorprogramm, das der Lieferant des Computers zur Verfügung stellt.

Dieses Programm ist üblicherweise in einem EPROM (oder PROM), dessen Inhalt der CPU sofort nach dem Start als Anweisung dient.

Es soll nicht Aufgabe dieses Buches sein, Monitorprogramme zu beschreiben und zu analysieren. An dieser Stelle sei nur ihr 'Lebenszweck' definiert.

Ein Monitorprogramm muß in einer CP/M-Maschine folgende Funktionen erfüllen können:

1. Nach dem Einschalten muß zumindest die Konsole (Ein- und Ausgabe) initialisiert werden.

2. Ein Speichertest ist von Nutzen falls unterschiedliche Speichergrößen möglich sind.

3. Das Gerät sollte melden, daß es für weitere Tätigkeiten bereit ist.

4. Der Monitor kann nun eine Vielzahl von Hilfsfunktionen zur Verfügung stellen oder auch nur eine kleine Auswahl von Sprungmöglichkeiten in andere Speicherbereich oder eben die Möglichkeit eine CP/M Diskette zu booten.

Es gibt Geräte, die anstelle des Monitorprogrammes ein Programm eingebunden haben, das sofort nach dem Einschalten versucht die Diskette in Laufwerk A zu booten. In diesem Falle muß alle Initialisierung, soweit nicht doch schon vorher geschehen, vom LDRBIOS oder BIOS übernommen werden.

Gleichgültig wie das Gundprogramm aussieht um CP/M 3 booten zu können, ist eine von zwei Möglichkeiten gegeben:

1. Sektor 1 auf Track 0 der Boot-Diskette wird (egal wie), z.B. nach Adresse 0000H in Bank 0 gelesen und danach diese Adresse angesprungen (d.h. dieses Programm wird ausgeführt).

2. Es wird der CPM-Lader nach Adresse 100H Bank 0 direkt geladen. Die Information, wo sich dieser Lader befindet, muß dem Boot-Programm bekannt sein.

Beide Arten des Bootens müssen die Möglichkeit bieten Boot-Fehler zu melden, im Sinne von: "Lesefehler" oder "Diskette nicht vorhanden".

Im ersten Fall werden, je nach physikalischer Sektorgröße, bis zu 1k-Byte gelesen. In diesem gelesenen Programm kann nun eine weitere Initialisierung vorgenommen werden. Aber die Hauptaufgabe des geladenen Programmes ist es, das zu tun, was im zweiten Fall getan würde, es wird der eigentliche Lader (CPMLDR) nach Adresse 100H in Bank 0 geladen.

Man fragt sich unwillkürlich, wozu dieser umständliche Weg im

ersten Fall gut sei, nun: Ein derartiger Booter kann unterschiedliche Betriebsysteme z.B. CP/M 2 und CP/M 3 laden und dazu vielleicht auch noch ein UCSD-Pascal System oder ein FORTH-System. Alle Informationen hierzu würden sich immer im ersten Sektor einer Diskette befinden. Der zweite Fall kann hier nicht (so ohne weiteres) benutzt werden, da die Größe (Länge) des zu ladenden Programmes sehr unterschiedlich sein kann.

Das nachfolgende Listing zeigt einen (sehr) einfachen Bootlader wie er z.B. beim NDR-Klein-Computer eingesetzt werden könnte.

(Es stehen dafür jedoch 'richtige' Monitorprogramme zur Verfügung)

```
                      ; ***************************************************************
                      ; *                                                             *
                      ; * MODUL               Boot-Lader                              *
                      ; * REV 1.0             850110                                  *
                      ; *                                                             *
                      ; * Aufgabe dieses Programmes ist es Sektor 1 Track 0 einer     *
                      ; * 5.25" Diskette in Laufwerk A nach Adresse 0 zu laden und    *
                      ; * dort auszufuehren.                                          *
                      ; *                                                             *
                      ; ***************************************************************

      0000                    org 0

0000  F3          boot:       di                        ; keine Interrupts
0001  21 000F                 ld      hl,@start         ; nun Programmteil 1 transferieren
0004  11 FC00                 ld      de,0fc00h         ; Zieladresse
0007  01 004A                 ld      bc,@len           ; Transferlaenge
000A  ED B0                   ldir
000C  C3 FC00                 jp      0fc00h            ; und dieses Programmsegment anspringen

      000F        @start      equ     $                 ; Transfersegment Start

                              .phase  0fc00h            ; Ausfuehrungsadresse

FC00  31 0000                 ld      sp,0              ; setzte eigene Stack
FC03  3E 80                   ld      a,10000000b       ; Bank 0 und Bootkarte schliessen
FC05  D3 C8                   out     (0C8H),a          ; EPROM zu RAM eingeschaltet

                              ;
                              ; das System hat jetzt 64k RAM in Bank 0 und das EPROM
                              ; der Bank Bootkarte ist abgeschaltet
                              ;

FC07  3E D0                   ld      a,11010000b       ; Befehl force interrupt
FC09  D3 C0                   out     (0C0H),a          ; an den FDC Kontroller zum RESET
FC0B  10 FE                   djnz    $                 ; kurz warten ( <B> war NULL nach DJNZ)
FC0D  3E 21                   ld      a,00100001b       ; select code MINIS Laufwerk A
FC0F  D3 C4                   out     (0C4H),a          ; ausgeben
FC11  10 FE                   djnz    $                 ; nochmals kurz warten
FC13  3E 09                   ld      a,00001001b       ; home Befehl mit 6 ms
FC15  D3 C0                   out     (0C0H),a          ; an FDC Kontroller
FC17  E3                      ex      (sp),hl           ; ca 20 ys warten
FC18  E3                      ex      (sp),hl
FC19  E3                      ex      (sp),hl
FC1A  E3                      ex      (sp),hl
```

```
FC1B  DB C0       busy:   in      a,(0C0H)        ; dann Status abfragen
FC1D  0F                  rrca                    ; Bit 7 ins Carry
FC1E  38 FB               jr      c,busy          ; nicht bereit solange Bit 7=1

                          ;
                          ; in diesem Treiber wird nun 'unendlich' oft gelesen
                          ; solange bis es klappt
                          ;

FC20  0E C3               ld      c,0C3H          ; Adresse des Datenregisters laden
FC22  AF          retry:  xor     a               ; A=0
FC23  D3 C1               out     (0C1H),a        ; ins Trackregister  (Track=0)
FC25  3C                  inc     a               ; A=1
FC26  D3 C2               out     (0C2H),a        ; ins Sektorregister (Sektor 1)
FC28  21 0000             ld      hl,0            ; Zieladresse
FC2B  3E 8C               ld      a,10001100b     ; Befehl Sektor lesen
FC2D  D3 C0               out     (0C0H),a        ; an den FDC-Kontroller
FC2F  E3                  ex      (sp),hl
FC30  E3                  ex      (sp),hl
FC31  E3                  ex      (sp),hl
FC32  E3                  ex      (sp),hl         ; 20 us warten
FC33  DB C0       loop:   in      a,(0C0H)        ; Status einlesen
FC35  CB 4F               bit     1,a             ; DRQ? (Daten liegen bereit?)
FC37  28 05               jr      z,loop1         ; wenn nicht sonst
FC39  ED A2               ini                     ; Daten einlesen
FC3B  C3 FC33             jp      loop            ; ... jedes Zeichen einzeln

FC3E  CB 47       loop1:  bit     0,a             ; fertig ?
FC40  C2 FC33             jp      nz,loop         ; wenn nicht
FC43  E6 BC               and     10111100b       ; sonst Fehler maskieren
FC45  20 DB               jr      nz,retry        ; wenn falsch gelesen nochmal

                          ;
                          ; hierhier wenn Sektor fehlerfrei gelesen wurde
                          ;

FC47  C3 0000             jp      0               ; Programm ausfuehren

                          .dephase

        004A      @len    equ     $-@start

                          end
```

Kapitel 3 Der Track-0-Lader

Das vom Boot-Lader geladene Programmsegment 'Track-0-Lader' hat die alleinige Aufgabe (zumindest in dieser Zusammenstellung) das CP/M Boot-System, den sog. CPM-Lader (CPMLDR) in den Arbeitsbereich des Computers (ab Adresse 100H) zu laden.

Dieser Track-0-Lader ist nun bereits CP/M 3 typisch, könnte für CP/M 2 jedoch ähnlich aussehen (dort muß jedoch auch aus Track 1 und evtl. noch Track 2 geladen werden) und für ein FORTH-System wiederum ganz anders.

Die Entscheidung, wie-was-wo gemacht wird, liegt meist in der Hand des System-Programmierers (das ist der, der Betriebssysteme wie CP/M in einem Computer zum 'laufen' bringt, d.h. die Hardware-Anpassung vornimmt - oder auch ein neues Betriebssystem schreibt).

Im Falle des NDR-Klein-Computers entspricht der Track-0-Lader dem nachfolgenden Listing. Auf eines sei bei dieser Gelegenheit hingewiesen: in der Zusammenstellung mit CP/M 3 gibt es einen speziellen Boot-Monitor mit einem Hardware-angepaßten Ein- Ausgabe-Teil ab Adresse FC00H.

Aus systemspezifischen Gründen (vor allem des Konsolentreibers wegen, der mit rund 4K sonst zu Buche schlagen würde) werden Routinen aus diesem Programmsegment, das immer resident bleibt, in allen Lader- und BIOS-Programmen mit verwendet. Ein Listing dieses Monitorsegmentes ist in Teil VI ausgedruckt, um entsprechende Aufrufe 'verfolgen' zu können.

```
                              MACLIB  DEFAULT.INC
                        ; ***************************************************************
                        ; *                                                             *
                        ; *  Track-0-Lader fuer NDR-LDRBIOS CP/M 3                      *
                        ; *                                                             *
                        ; * Dieser Lader wird vom BOOT-Lader nach SECSEG geladen (F000H) *
                        ; * und dort ausgefuehrt.                                       *
                        ; * Um Programme laden zu koennen, die diesen Beeich ueberladen *
                        ; * ist es notwendig den eigentlichen Lader nach Adresse 0 um-  *
                        ; * zuladen.                                                    *
                        ; *                                                             *
                        ; * Geschrieben von RAOUL O. KOERBER, Detmold                   *
                        ; *                                                             *
                        ; ***************************************************************

                              .phase 0f000h           ; Puffer unter ELMON

F000   21 F00C                ld      hl,getit        ; Transfer nach Adresse 0
F003   11 0000                ld      de,0            ; des geladenen Sektors
F006   01 003B                ld      bc,len
F009   ED B0                  ldir
F00B   C7                     rst     0
          F00C      getit     equ     $

```

```
                                .dephase
                                .phase 0

0000  11 0100       trk0:   ld      de,100h         ; dorthin
0003  21 F080               ld      hl,0f080h       ; die ..
0006  01 0380               ld      bc,7*80h        ; naechsten 7 Sektoren
0009  ED B0                 ldir
000B  EB                    ex      de,hl           ; neue DMA-Adresse nach HL
000C  01 0121               ld      bc,0121h        ; lesen von Laufwerk 'A'
000F  11 0002               ld      de,0002h        ; ab Sektor 2
0012  C5            trk1:   push    bc
0013  D5                    push    de
0014  E5                    push    hl
0015  CD FC27               call    efloppy         ; naechsten Sektor lesen
0018  C4 0034               call    nz,error
001B  E1                    pop     hl              ; DMA
001C  11 0400               ld      de,400h         ; +1k
001F  19                    add     hl,de           ; gibt neue DMA-Adresse
0020  D1                    pop     de
0021  C1                    pop     bc
0022  1C                    inc     e               ; Sektor+1
0023  7B                    ld      a,e
0024  FE 06                 cp      6               ; 5 Sektoren geladen?
0026  38 EA                 jr      c,trk1          ; weiter wenn nicht
0028  3A 0033               ld      a,(erflg)       ; Fehler aufgetreten?
002B  B7                    or      a
002C  CA 0100               jp      z,100h          ; wenn nicht
002F  37                    scf
0030  C3 F400               jp      0f400h          ; zurueck in Monitor

0033  00            erflg:  db      0

0034  E5            error:  push    hl
0035  21 0033               ld      hl,erflg
0038  34                    inc     (hl)
0039  E1                    pop     hl
003A  C9                    ret

        003B        len     equ     $-trk0

                            .dephase

                            end
```

Kapitel 4 Der CPM-Lader (CPMLDR)

Dieses Programmsegment von CP/M 3 hat nur eine einzige Aufgabe, die Datei CPM3.SYS zu laden und auf die richtigen Adressen in Bank 0 (dem gebankten Bereich von CP/M 3) und in den sog. gemeinsamen (Common) Bereich (von E000..FFFF) zu verteilen.

Man muß sich die genannte Speicheraufteilung so vorstellen, daß, solange Bank 0 eingeschaltet ist, das komplette CP/M mit LOADER, BDOS und BIOS in einer Speicherebene vorliegt, da auf den gemeinsamen Bereich ja eben von allen Banken aus zugegriffen werden kann.

Diese Speicher-Konfiguration liegt immer dann vor (vom BDOS erzwungen), wenn ein BDOS-Aufruf Programmteile benötigt, die im gebankten Teil (Bank 0) angesiedelt sind. Dies trifft (vom BDOS aus gesehen) auch auf den gebankten Teil des BIOS zu.

Der CPM-Lader wird von Track-0-Lader nach Bank 0 Adresse 100H geladen und angesprungen. Vom Prinzip her ist der CPM-Lader ein 'normales' CP/M mit Sonderaufgaben. (Es 'weiß', wie die Datei CPM3.SYS - das eigentliche CP/M3 - zu verteilen ist, kann aber keine Dateien auf Diskette schreiben, weiß auch nichts mit Uhrzeit und Bankumschaltung anzufangen).

Dieser CPM-Lader besteht nun - um die Sache recht kompliziert erscheinen zu lassen - wiederum aus zwei Teilen, dem Hardware-unabhängigen BDOS-Teil (Basic Disk Operating System - dem grundlegenden Disketten Betriebssystem) und der Hardware-Anpassung, dem Lader-BIOS.

Das BIOS kann recht einfach aufgebaut sein, da nur wenige Funktionen benötigt werden, es entspricht im übrigen jedoch einem 'normalen' BIOS wie das nachfolgende Listing zeigt.

Die Einzelheiten des LDRBIOS-Listing sollen an dieser Stelle unbesprochen bleiben. Alle (notwendigen) Einzelheiten werden bei der Besprechung der Segmente des kompletten BIOS erklärt.

Zur Generierung des gesamten CPM-Laders wird der BDOS-Anteil von Digital Research als REL-Datei CPMLDR.REL mitgeliefert. (Lader) BDOS, (Lader)BIOS und Track-0-Lader werden nun wie auf der folgenden Seite gezeigt, zusammengebunden.

Die abgespeicherte Datei wird mit dem Index .BIN versehen, zur Erkennung, daß das Programm zwar BINär ist, aber nicht als COM-Programm ablauffähig ist, da der Track-0-Lader in diesem Falle bei Adresse 0 beginnt.

Diese Datei muß nun mit einem Hilfsprogramm auf Track 0 ab Sektor 1 geschrieben werden. Hierzu gibt es die unterschiedlichsten Hilfsmittel mit den unterschiedlichsten Möglichkeiten. Das Monitorprogramm ELMON für den NDR-Klein-Computer bietet hierzu hervorragende Unterstützung. Zur Generierung der Programme ist jedoch der Zugriff auf ein beliebiges CP/M-System erforderlich - hier beißt sich dann eben doch die Katze in den Schwanz.

```
A>m80 =track0                   < assemblieren des Track0-Laders
A>m80 =ldrbios                  < assemblieren des Lader-Bios
A>link track0[l0000]            < linken des Track-0 Laders auf Adresse 0000H
A>link cpmldr[l100]=cpmldr,ldrbios < linken des CPM-Laders mit dem BIOS

A>sid track0.com                < laden des Track0-Laders mit SID
CP/M 3 SID - Version 3.0        < so meldet sich SID
NEXT MSZE  PC  END
0180 0180 0100 C9FF             < Start bei 100H benutzter Speicher bis 180H

#d100                           < Inhalt ueberpruefen (dump ab 100h)


0100: 21 0C F0 11 00 00 01 3B 00 ED B0 C7 11 00 01 21 !......;.......!
0110: 80 F0 01 80 03 ED B0 EB 01 21 01 11 02 00 C5 D5 .........!......
0120: E5 CD 27 FC C4 34 00 E1 11 00 04 19 D1 C1 1C 7B ..'..4.........{
0130: FE 06 38 EA 3A 33 00 B7 CA 00 01 37 C3 00 F4 00 ..8.:3.....7....
0140: E5 21 33 00 34 E1 C9 1A 1A 1A 1A 1A 1A 1A 1A 1A .!3.4...........
0150: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0160: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0170: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0180: 43 50 2F 4D 20 33 20 53 49 44 20 2D 20 56 65 72 CP/M 3 SID - Ver
0190: 73 69 6F 6E 20 33 2E 30 24 31 00 02 C5 C5 11 80 sion 3.0$1......
01A0: 01 0E 09 CD 05 00 C1 21 07 00 7E 3D 90 57 1E 00 .......!..~=.W..
01B0: D5 21 00 02 78 B1 CA C1 01 0B 7E 12 13 23 C3 B4 .!..x.....~..#..


#rcpmldr.com,80                 < laden von CPMLDR.COM nach Adresse 180H!
NEXT MSZE  PC  END
0080 0D80 0100 C9FF             < Programmende bei D80H

#d100                           < Inhalt ueberpruefen


0100: 21 0C F0 11 00 00 01 3B 00 ED B0 C7 11 00 01 21 !......;.......!
0110: 80 F0 01 80 03 ED B0 EB 01 21 01 11 02 00 C5 D5 .........!......
0120: E5 CD 27 FC C4 34 00 E1 11 00 04 19 D1 C1 1C 7B ..'..4.........{
0130: FE 06 38 EA 3A 33 00 B7 CA 00 01 37 C3 00 F4 00 ..8.:3.....7....
0140: E5 21 33 00 34 E1 C9 1A 1A 1A 1A 1A 1A 1A 1A 1A .!3.4...........
0150: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0160: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0170: 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A 1A ................
0180: 31 81 02 CD 00 0B 0E 0D CD 8D 02 0E 09 11 25 02 1.............%.
0190: CD 8D 02 0E 0F 11 AB 01 CD 8D 02 FE FF 11 CF 01 ................
01A0: CA A2 01 11 80 00 CD 8F 01 CD 95 01 21 80 00 11 ............!...
01B0: 81 02 0E 06 7E 12 13 23 0D C2 34 01 CD 95 01 0E ....~..#..4.....

< Bei Adresse 180H beginnt nun der CP/M-Lader . Dies entspricht spaeter >
< dem zweiten logischen Sektor (je 128 = 80H Bytes pro Sektor )          >

#wcpload.bin,100,d80            < Programm als cpload.bin abspeichern

0019h record(s) written.        < Bestaetigung des Abspeicherns 19 Sektoren

#g0                             < Warmstart
```

```
                    MACLIB  DEFAULT.INC      ; Vereinbarungen
                    MACLIB  CPM3.INC         ; und Macros
                ; ***********************************************************
                ; * LDRBIOS - Lader Bios fuer CP/M3 und NDR-Computer         *
                ; * Systemvoraussetzung:                                     *
                ; * Vollausbau-Z80-CPU, 128k DRAM, FLO2, SER-Karte als Terminal- *
                ; * Anschluss oder GDP und KEY                               *
                ; *                                                          *
                ; * Aufgabe des LDRBIOS ist es, die Datei CPM3.SYS zu laden  *
                ; * - CPM3.SYS ist das konfigurierte Benutzerspezifische CPM3 *
                ; *                                                          *
                ; * Version 1.0 vom 2.1.1985                                 *
                ; * geschrieben von Raoul O. Koerber im Auftrage des         *
                ; * ELEKTRONIKLADEN DETMOLD                                  *
                ; ***********************************************************

                    ;
                    ; ***************
                    ; Bios-Sprungtabelle
                    ; ***************
                    ;

0000' C3 008B'  ?BOOT:  jp      boot            ; Kaltstart
0003' C3 008B'  ?WBOOT: jp      error           ; Fehler wenn Warmstart
0006' C3 FC00   ?CONST: jp      econist         ; Benutze hierzu
0009' C3 FC03   ?CONIN: jp      econin          ; MONIO aus ELMON

000C' C3 FC06   ?CONO:  jp      econout
000F' C3 008B'  ?LIST:  jp      error
0012' C3 008B'  ?AUXO:  jp      error
0015' C3 008B'  ?AUXI:  jp      error

0018' C3 0050'  ?HOME:  jp      .home           ; Laufwerk auf Track 0 setzen
001B' C3 008C'  ?SLDSK: jp      seldsk          ; Ansprache Laufwerk und Parameteruebergabe
001E' C3 0093'  ?STTRK: jp      settrk          ; Track setzen
0021' C3 0098'  ?STSEC: jp      setsec          ; Sektor setzen

0024' C3 009D'  ?STDMA: jp      setdma          ; DMA-Adresse setzen
0027' C3 00B2'  ?READ:  jp      read            ; lesen 1 physikalischen Blocks
002A' C3 008B'  ?WRITE: jp      error           ; kein Schreiben im Lader
002D' C3 008B'  ?LISTS: jp      error

0030' C3 00A7'  ?SCTRN: jp      sectrn          ; Skew-Faktor uebersetzen
0033' C3 008B'  ?CONOS: jp      error
0036' C3 008B'  ?AUXIS: jp      error
0039' C3 008B'  ?AUXOS: jp      error

003C' C3 008B'  ?DVTBL: jp      error
003F' C3 008B'  ?DEVIN: jp      error
0042' C3 008B'  ?DRTBL: jp      error
0045' C3 008B'  ?MLTIO: jp      error

0048' C3 008B'  ?FLUSH: jp      error
004B' C3 00A2'  ?MOV:   jp      move            ; Transfer von Daten


                    ; Rest wird nicht benoetigt -----

```

```
                          ;
                          ; ****************
                          ; XDPH
                          ; ****************

004E'  00                 DB      00B             ; Steppingrate 00=3ms 01=6ms
004F'  21                 DB      021H            ; Laufwerksselekt
0050'  0086'      xdph:   DW      xlt             ; Sektor Skew
0052'  0000               DW      000H
0054'  0000               DW      000H
0056'  0000               DW      000H
0058'  0000               DW      000H
005A'  0000               DW      000H            ; MEDIA FLAG
005C'  0075'      setdpb: DW      DPBT80D         ; von LOGIN setzten wenn mehrer moeglich
005E'  0000               DW      000H            ; CHECKSUM VECTOR
0060'  0000               DW      000H            ; ALLOC VECTOR
0062'  0069'              DW      DIRBCB          ; DIRECTORY BUFFER CONTROLL BLOCK
0064'  0069'              DW      DTABCB          ; DATA BUFFER CONTROLL BLOCK
0066'  FFFF               DW      0FFFFH          ; HASH TABELLE DISABLE
0068'  00                 DB      0               ; HASH BANK

0069'             DTABCB:
0069'  FF         DIRBCB: DB      0FFH            ; DRIVE CODE
006A'  00                 DB      000H
006B'  00                 DB      000H
006C'  00                 DB      000H
006D'  00                 DB      000H            ; WRITE FLAG
006E'  00                 DB      000H            ; SCRATCH
006F'  0000               DW      000H            ; TRACK
0071'  0000               DW      000H            ; SECTOR
0073'  01B1'              DW      HBUFF           ; BUFFER


                          ;
                          ; ****************
                          ; DPB
                          ; ****************
                          ;

0075'             dpbt80d:dpb     1024,5,160,2048,256,4 ; an andere Laufwerke Anpassen!!!!
0075'  0028       A       defw    ??0001                ; 128 Byte (log)-Sektoren pro Track
0077'  04 0F      A       defb    ??0002,??0003         ; Block-shift und Maske
0079'  00         A       defb    ??0004                ; Extent-Mmaske
007A'  0185       A       defw    ??0005                ; Maximale Block-Anzahl
007C'  00FF       A       defw    ??0006                ; Maximale DIR-Eintraege
007E'  F0 00      A       defb    ??0007,??0008         ; Belegungs-Vektoren
0080'  0040       A       defw    ??0009                ; Pruefsumme
0082'  0004       A       defw    4                     ; Reservierte Tracks (System)
0084'  03 07      A       defb    ??000A,??000B         ; (phys)-Sektor Groesse- &  Shift-Maske

0086'              xlt:   skew    5,1,1
0086'  01         B       defb    ?nxtsec+1
0087'  02         B       defb    ?nxtsec+1
0088'  03         B       defb    ?nxtsec+1
0089'  04         B       defb    ?nxtsec+1
008A'  05         B       defb    ?nxtsec+1
```

```
                        ;
                        ; ****************
                        ; Laderstart
                        ; ****************
                        ;

0088'           boot:

                        ;
                        ; Hier kann CPM-Typische Initialisierung erfolgen
                        ; die vom Bootprogramm (ELMON) nicht durchgefuehrt
                        ; wird, vom CPM-Lader jedoch benoetigt wird
                        ;

008B' C9        error:  ret                     ; vorl nichts ...

008C'           seldsk:

                        ;
                        ; Es wird die Adresse des XDPH's des gebooteten Laufwerkes
                        ; in <HL> zurueckerwartet
                        ;

008C' 21 0050'          ld      hl,xdph
008F' C9                ret

                        ;
                        ; nachfolgende Daten werden vorlaeufig nur initialisiert
                        ; bzw abgelegt. Aktion erfolgt erst beim aktiven Lesen
                        ;

0090' 01 0000    home:  ld      bc,0            ; ... entspricht Track0


0093' ED 43 01AB' settrk: ld    (@trk),bc       ; Track Nummer ablegen
0097' C9                ret

0098' ED 43 01AD' setsec: ld    (@sect),bc      ; Sektor Nummer ablegen
009C' C9                ret

009D' ED 43 01AF' setdma: ld    (@dma),bc       ; gueltige Adresse ablegen
00A1' C9                ret

00A2'            move:

                        ;
                        ; Transfer von Speicherinhalten mit Z80 Power
                        ; Uebergabe <HL>=Ziel,<DE>=Quelle,BC=Laenge
                        ;

00A2' EB                ex      de,hl           ; Ziel in <DE> Quelle in <HL>
00A3' ED B0             ldir                    ; fuer Transfer mit Z80-Power
00A5' EB                ex      de,hl           ; Register in 'Originalreihenfolge' zurueck
00A6' C9                ret

```

```
00A7'               sectrn:

                           ;
                           ; Uebersetzung des Skews
                           ; Adresse der Uebersetzungstabelle in <DE>, ist <DE>=0
                           ; ist KEINE Uebersetzung notwendig, die logische Sektoradresse
                           ; in <BC> entspricht dann der physikalischen Sektoradresse
                           ;

00A7' 69                   ld      l,c            ; Kopie von <BC> nach <HL>
00A8' 60                   ld      h,b            ;  falls kein SKEW
00A9' 7A                   ld      a,d            ; wenn <DE>=0 kein SKEW
00AA' B3                   or      e
00AB' C8                   ret     z              ; log Sektor=phys Sektornummer
00AC' EB                   ex      de,hl          ; <DE> -> <HL> um
00AD' 09                   add     hl,bc          ; Offset in Tabelle berechnen
00AE' 6E                   ld      l,(hl)         ; phys Sektor Numme nach <HL>
00AF' 26 00                ld      h,0            ; die physikalsiche Sektoradresse
00B1' C9                   ret                    ; mit phys. Sektoradresse in <HL>


00B2'               read:

                           ;
                           ; lesen eines Sektors
                           ; Aufruf mit folgenden Argumenten bereits gesetzt:
                           ; Laufwerksnummer in (@drv)y
                           ; Transferadresse in (@dma)
                           ; Transferbank    in (@dbnk)
                           ; Tracknummer     in (@trk)
                           ; Sektornummer    in (@sect)
                           ; Zeiger auf XDPH in   <DE>
                           ;

00B2' 01 0179'             ld      bc,read$sector
00B5' C5                   push    bc             ; rette Funktion
00B6' CD 0136'             call    setup          ; berechne phys Track setze sso
00B9' E1                   pop     hl             ; Adresse des UP's (READ-WRITE)

00BA'               more$retries:

00BA' 06 0A                ld      b,10           ; 10 Versuche zulassen

00BC'               retry$loop:

00BC' C5                   push    bc             ; Schleifenzaehler
00BD' E5                   push    hl             ; UP
00BE' 3A 019F'             ld      a,(cursel)     ; neuer select (ohne sso)
00C1' 21 01A3'             ld      hl,oldsel      ; Zeiger auf 'letzen' select
00C4' BE                   cp      (hl)
00C5' 77                   ld      (hl),a         ; fuer's naechste mal ...
00C6' 20 0A                jr      nz,new$track   ; dann SEEK
00C8' 3A 01AB'             ld      a,(@trk)       ; alter Track (logisch)
00CB' 21 01A2'             ld      hl,oldtrk      ; Zeiger auf 'letzen' Track (logisch)
00CE' BE                   cp      (hl)
00CF' 77                   ld      (hl),a
00D0' 28 09                jr      z,same$track

```

```
00D2'                  new$track:

00D2'  CD 00FB'                  call    check$seek      ; Track einstellen
00D5'  01 0064                   ld      bc,100          ; 100ms
00D8'  CD 016C'                  call    delay           ; warten


00DB'                  same$track:

00DB'  3A 01A1'                  ld      a,(curtrk)      ; physikalischer Track
00DE'  D3 C1                     out     (fdctrk),a      ; setzen
00E0'  3A 01AD'                  ld      a,(@sect)       ; physikalischen Sektor
00E3'  D3 C2                     out     (fdcsec),a      ; setzen
00E5'  E1                        pop     hl              ; UP
00E6'  E5                        push    hl              ; ... retten fuer weiter Versuche
00E7'  CD 019E'                  call    ipchl           ; ausfuehren
00EA'  E1                        pop     hl              ;
00EB'  C1                        pop     bc              ; Schleifenzaehler
00EC'  C8                        ret     z               ; Fehlerfrei ...
00ED'  C5                        push    bc
00EE'  E5                        push    hl
00EF'  CB 67                     bit     4,a             ; evtl RNF-Fehler
00F1'  C4 00FB'                  call    nz,check$seek   ; dann nochmals SEEK
00F4'  E1                        pop     hl              ; UP
00F5'  C1                        pop     bc              ; Schleifenzaehler
00F6'  10 C4                     djnz    retry$loop      ; ... auch nach nochmaligem SEEK

00F8'                  hard$error:

00F8'  3E 01                     ld      a,1
00FA'  C9                        ret

                                 ;
                                 ; ***************
                                 ; Hilfsprogramme
                                 ; ***************
                                 ;

00FB'                  check$seek:

00FB'  CD 012A'                  call    read$id         ; versuche ID-Feld zu lesen
00FE'  28 05                     jr      z,id$ok         ; ... wenn ok
0100'  CD 0114'                  call    restore         ; sonst HOME
0103'  06 00                     ld      b,0
0105'  78              id$ok:    ld      a,b
0106'  D3 C1                     out     (fdctrk),a      ; ins Trackregister
0108'  21 01A1'                  ld      hl,curtrk       ; Zeiger auf physikalischen Track
010B'  BE                        cp      (hl)            ; beide gleich ?
010C'  C8                        ret     z               ; dann fertig
010D'  7E                        ld      a,(hl)          ; sonst SOLL-Register
010E'  D3 C3                     out     (fdcdat),a      ; ins Datenregister
0110'  06 10                     ld      b,seek          ; + SEEK Befehl
0112'  18 02                     jr      busy$cmd        ; ausfuehren


```

```
0114'                restore:

0114' 06 00                  ld      b,home          ; Laufwerk auf TRACK 0

0116'                busy$cmd:

0116' 3A 01A0'               ld      a,(curstep)     ; stepping-rate
0119' B0                     or      b               ; einfuegen
011A' D3 C0                  out     (fdccmd),a      ; Befehl ausgeben
011C' DB C0          busy:   in      a,(fdccmd)
011E' 0F                     rrca
011F' 30 FB                  jr      nc,busy
0121' DB C0          busy1:  in      a,(fdccmd)      ; nun abwarten
0123' CB 47                  bit     0,a             ; bis NICHT mehr busy
0125' 20 FA                  jr      nz,busy1
0127' E6 90                  and     10010000b       ; maskiere READY & RNF-SEEK
0129' C9                     ret

012A'                read$id:

                             ;
                             ; lesen des ID-Feldes in den IDF-Puffer
                             ;

012A' 21 01A5'               ld      hl,idfbuf       ; Braucht eigenen Puffer
012D' 06 C4                  ld      b,readid
012F' E5                     push    hl
0130' CD 017E'               call    getdata
0133' E1                     pop     hl
0134' 46                     ld      b,(hl)          ; Tracknummer
0135' C9                     ret

0136' 21 004F'       setup:  ld      hl,xdph-1       ; Select
0139' 5E                     ld      e,(hl)          ; lesen
013A' 2B                     dec     hl              ; Steppingrate
013B' 56                     ld      d,(hl)
013C' ED 53 019F'            ld      (cursel),de     ; ablegen

                             ;
                             ; physikalischen Track aus logischen Track berechnen
                             ; und damit side-select
                             ;

0140' 3A 01AB'               ld      a,(@trk)        ; zuvor logischen TRACK einlesen
0143' CB 7B                  bit     7,e             ; einseitiges Laufwerk?
0145' 20 06                  jr      nz,setx         ; wenn JA
0147' CB 6B                  bit     5,e             ; evtl 8" sd?
0149' 28 02                  jr      z,setx          ; wenn JA
014B' B7                     or      a               ; reset CARRY
014C' 1F                     rra                     ; /2
014D' 32 01A1'       setx:   ld      (curtrk),a      ; ist physikalischer Track
0150' 3E 00                  ld      a,0             ; sso-Flag schuetzen ...
0152' 30 04                  jr      nc,setup1       ; Seite 0(1)
0154' CB CF                  set     1,a             ; sso-bit setzen
0156' CB FB                  set     7,e             ; auch in SELCOD side-select einbringen
0158' 32 01A4'       setup1: ld      (ssoflag),a     ; sso-Flag ablegen
015B' 7B                     ld      a,e             ; select
015C' D3 C4                  out     (fdcsel),a      ; ausgeben
```

```
                                ;
                                ; Testen ob Diskette eingelegt ist
                                ;

015E' 21 0000       repeat: ld      hl,0            ; time_out
0161' 2B            setup2: dec     hl              ; time_out -1
0162' 7C                    ld      a,h
0163' B5                    or      l
0164' 28 F8                 jr      z,repeat        ;  es kann nicht sein .....
0166' DB C0                 in      a,(fdccmd)
0168' 07                    rlca                    ; READY?

                                ;
                                ; Anmerkung: bei festverdrahtetem READY ist diese Routine
                                ; sinnlos !!!! dann evtl Index-Loch abfragen ... nach 250ms Delay
                                ;

0169' 38 F6                 jr      c,setup2
016B' C9                    ret

016C'               delay:

                                ;
                                ; Warteschleife mit Zaehler in <BC>
                                ; Je einheit wird 1 ms gewartet
                                ;

016C' 0B                    dec     bc
016D' C5                    push    bc
016E' 06 E6                 ld      b,230           ; bei 4 MHz
0170' 00            del1:   nop
0171' 10 FD                 djnz    del1
0173' C1                    pop     bc
0174' 78                    ld      a,b
0175' B1                    or      c
0176' 20 F4                 jr      nz,delay
0178' C9                    ret

0179'               read$sector:

                                ;
                                ; lesen eines physikalischen Sektors
                                ; Erwartet Disk-Track-Sektor-DMA gesetzt
                                ;

0179' 06 88                 ld      b,reads
017B' 2A 01AF'              ld      hl,(@dma)

017E' 3A 01A4'      getdata:ld      a,(ssoflag)     ; sso-Bit
0181' B0                    or      b               ; in Befehl einfuegen
0182' 0E C3                 ld      c,fdcdat        ; Datenport
0184' D3 C0                 out     (fdccmd),a      ; Befehl ausgeben
0186' DB C4         getd1:  in      a,(fdcsel)      ; abwarten bis
0188' 07                    rlca                    ; BUSY aktiv
0189' 30 FB                 jr      nc,getd1
018B' DB C0         getd2:  in      a,(fdccmd)      ; Status abfragen
018D' CB 4F                 bit     1,a             ; DRQ?
018F' 28 05                 jr      z,getd3         ; wenn noch nicht ...
0191' ED A2                 ini                     ;   sonst Daten einlesen
0193' C3 018B'              jp      getd2           ; und weiter abfragen

0196' CB 47         getd3:  bit     0,a             ; nicht mehr BUSY?
0198' C2 018B'              jp      nz,getd2
```

```
                                ;
                                ; gelesen ...
                                ;

     019B' E6 FC                and     11111100b       ; Fehler maskieren
     019D' C9                   ret

     019E' E9           ipchl:  jp      (hl)            ; indirekter CALL

                                ;
                                ; ****************
                                ; Variablenbereich
                                ; ****************
                                ;

     019F'   0001       cursel: ds      1               ; momentaner Select
     01A0'   0001       curstep:ds      1               ; momentane stepping-rate
     01A1'   0001       curtrk: ds      1               ; physikalischer Sektor
     01A2'   0001       oldtrk: ds      1
     01A3' FF           oldsel: db      -1              ; ungueltig beim Start
     01A4'   0001       ssoflag:ds      1
     01A5'   0006       idtbuf: ds      6
     01AB' 0000         @trk:   dw      0
     01AD' 0000         @sect:  dw      0
     01AF' 0000         @dma:   dw      0


             01B1'      hbuff   equ     $

                                end
```

Kapitel 5 Das CP/M BIOS - Ein Überblick

Das BIOS (Basic Input Output System - Grundsystem der Datenein- und Ausgabe) ist die Hardware-Anpassung an eine bestimmte CP/M-Maschine.

Prinzipiell unterscheidet sich das BIOS verschiedener Computer nur im Aufbau bestimmter Hardware-Treiber. Dieser Unterschied kann sehr gering, aber auch recht drastisch sein, ist jedoch immer nur abhängig von der Ähnlichkeit der benutzten Hardware.

Immer gleich ist auf jeden Fall die Einsprungleiste des BIOS. Hierunter ist eine Sprungtabelle in die verschiedensten Unterprogramme zu verstehen, über die das BDOS oder gegebenfalls auch ein TPA-Programm bestimmte Funktionen des BIOS erreichen kann.

Der Aufbau der Sprungtabelle, deren Reihenfolge unveränderbar ist, ist dem in Kapitel 6 abgedruckten BIOS-Kernprogramm (BIOSKRNL) zu entnehmen.

Die einzelnen Funktionen des BIOS werden auf den nachfolgenden Seiten etwas genauer beschrieben. Im Übrigen wird hierzu auf die abgedruckten Listings verwiesen.

```
+-------------------------------------------------------+
! BIOS Funktion 0    BOOT (Kaltstart)                   !
+-------------------------------------------------------+
! Initialisierung des CP/M 3 Systems und laden des      !
! CCP's von Diskette. Starten des Systems               !
+-------------------------------------------------------+
! Eingabe            keine                              !
! Zurück             in den CCP                         !
+-------------------------------------------------------+
```

Diese Einsprungstelle wird vom CP/M-Lader aus Bank 0 heraus angesprungen.

In diesem Unterprogramm ist die letzte Möglichkeit gegeben, Schnittstellen, die nicht vom Monitor, Boot-Programm oder dem Lader-BIOS angesprochen wurden, nachzuinitialisieren.

Nach der Grund-Initialisierung werden noch die Einsprungadressen in den Warmstart (WBOOT) auf Adresse 0 der TPA und der Einsprung in das BIOS auf Adresse 5 in der TPA initialisiert.

Danach wird die Datei CCP.COM von Diskette in die TPA und einen reservierten Speicherbereich geladen. Ebenfalls aus diesem Unterprogramm heraus wird die Systemmeldung auf die Konsole ausgegeben und erst danach die Kontrolle an den CCP übergeben.

Da aus diesem Programm heraus alle Parameter neu (bzw. erstmals) gesetzt werden, spricht man von einem Kaltstart. Er wird nur nach dem Booten vorgenommen.

Es ist notwendig nach Aufruf dieses Unterprogrammes einen eigenen Stackbereich im gemeinsamen Speicherbereich bereitzustellen. Aus dem Programm heraus können nach der endgültigen Hardware-Initialisierung BDOS-Aufrufe vorgenommen werden. Diese werden vor allem benötigt um den CCP von der BOOT-Diskette zu laden. Falls die Datei CCP.COM auf der Bootdiskette nicht gefunden wird, muß dies dem Benutzer gemeldet werden.

Dieses Unterprogramm kann teilweise in Bank 0 liegen.

BIOS Funktion 1	WBOOT (Warmstart)
Nachladen und Aufruf des CCP's	
Eingabe	keine
Zurück	in den CCP

Dieses Unterprogramm wird von jedem Sprung auf Adresse 0 in der TPA oder dem Ausführen der Funktion 0 als BDOS-Aufruf ausgeführt.

Das Unterprogramm übernimmt eine Neuinitialisierung der Systemparameter, jedoch nicht die der Hardware-(Nach-) Initialisierung.

Es erfolgt nach Aufruf des Unterprogrammes keine Systemmeldung, vielmehr wird der CCP aus dem dafür reservierten Speicherbereich in einer Bank, in die TPA umgeladen und dort direkt ausgeführt.

Die Funktion muß einen eigenen Stack bereitstellen, der im gemeinsamen Speicherbereich liegen muss.

Es ist zu beachten, daß weder der Warmstart noch der CCP die Disketten Parameter zurücksetzt. Dies kann nur mit einem CNTRL-C nach dem Prompter des CCP's erfolgen oder durch einen entsprechenden BDOS-Aufruf.

Dieses Unterprogramm liegt im gemeinsamen (Common) Speicherbereich.

```
+-------------------------------------------------------+
! BIOS-Funktion 2   CONST (Konsolenstatus)              !
+-------------------------------------------------------+
! Lesen des Eingabestatus der Konsole                   !
+-------------------------------------------------------+
! Eingabe             keine                             !
! Zurück              A=00  es liegt keine Eingabe vor  !
!             oder    A=FF  es liegt eine Eingabe vor   !
+-------------------------------------------------------+
```

Diese Funktion übergibt dem Aufrufer den Eingabestatus der Konsole. Dieses Unterprogramm liegt im gemeinsamen Speicherbereich, kann also aus allen Banken aufgerufen werden.

```
+-------------------------------------------------------+
! BIOS-Funktion 3   CONIN (Konsoleneingabe)             !
+-------------------------------------------------------+
! Lesen eines eingegebenen Zeichens von der Konsole     !
+-------------------------------------------------------+
! Eingabe             keine                             !
! Zurück              A= eingegebenes Zeichen           !
+-------------------------------------------------------+
```

Diese Funktion liest ein Zeichen von der Konsole (Tastatur) und übergibt es dem Aufrufer in Register A. Liegt kein Zeichen vor, so wird gewartet, bis ein Zeichen eingegeben wurde.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+-------------------------------------------------------+
! BIOS-Funktion 4   CONOUT (Konsolenausgabe)            !
+-------------------------------------------------------+
! Ausgabe eines Zeichens auf die Konsole (Bildschirm)   !
+-------------------------------------------------------+
! Eingabe             C= auszugebendes Zeichen          !
! Zurück              A= ausgegebenes Zeichen           !
+-------------------------------------------------------+
```

Diese Funktion gibt das in Register C übergebene Zeichen an die Konsole aus. Das Zeichen wird im ASCII-Format ohne Parity übergeben.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+-------------------------------------------------------+
! BIOS-Funktion 5   LIST (Druckerausgabe)               !
+-------------------------------------------------------+
! Ausgabe eines Zeichens auf den Drucker                !
+-------------------------------------------------------+
! Eingabe             C= auszugebendes Zeichen          !
! Zurück              A= ausgegebenes Zeichen           !
+-------------------------------------------------------+
```

Diese Funktion gibt das Zeichen in Register C auf den Drucker aus. Das Zeichen ist im ASCII-Format ohne Parity.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+-----------------------------------------------------------+
! BIOS-Funktion 6   AUXOUT ( Ausgabe auf AUX )              !
+-----------------------------------------------------------+
! Ausgabe eines Zeichens auf den AUX-Kanal                  !
+-----------------------------------------------------------+
! Eingabe            C= auszugebendes Zeichen               !
! Zurück             A= ausgegebenes Zeichen                !
+-----------------------------------------------------------+
```

Diese Funktion gibt das Zeichen in Register C an den Zusatzkanal (AUX) aus. Das Zeichen ist im ASCII-Format ohne Parity.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+-----------------------------------------------------------+
! BIOS-Funktion 7   AUXIN ( Eingabe von AUX)                !
+-----------------------------------------------------------+
! Einlesen eines Zeichen vom AUX-Kanal                      !
+-----------------------------------------------------------+
! Eingabe            keine                                  !
! Zurück             A= eingegebenes Zeichen                !
+-----------------------------------------------------------+
```

Diese Funktion liest ein Zeichen vom Zusatzkanal (AUX).

Von verschiedenen Programmen (z.B. PIP.COM) wird ein eingelesenes 1AH als Endezeichen einer Übertragung verstanden. Liegt beim Aufruf der Funktion noch kein Zeichen vor, wird auf die Zeicheneingabe gewartet.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+---------------------------------------------------------+
! BIOS-Funktion 8   HOME  (Track 0 )                      !
+---------------------------------------------------------+
! Lesekopf des Laufwerkes über Track 0 positionieren      !
+---------------------------------------------------------+
! Eingabe             keine                               !
! Zurück              nichts                              !
+---------------------------------------------------------+
```

Diese Funktion führt die Positionierung des Kopfes direkt aus oder setzt den Track-Zeiger auf 0. Im letzteren Falle wird die Positionierung erst beim nächsten Schreib- oder Lesezugriff vorgenommen.

Dieses Unterprogramm kann in Bank 0 liegen.

```
+---------------------------------------------------------+
! BIOS-Funktion 9   SELDSK ( Laufwerk anmelden )          !
+---------------------------------------------------------+
! Anmelden und selektieren eines Laufwerkes               !
+---------------------------------------------------------+
! Eingabe             C= anzusprechendes Laufwerk         !
!                        00=A, 01=B .... 0F=P             !
!                     E= Erstaufruf-Flag                  !
! Zurück              HL= Adresse des XDPH oder           !
!                     HL= 0000 falls ungültiges Laufwerk  !
+---------------------------------------------------------+
```

Diese Funktion selektiert das in Register C genannte Laufwerk für den weiteren Diskettenzugriff.

Die Selektion bleibt bis zum nächsten (geänderten) Aufruf erhalten. Bei jedem Aufruf muß die Funktion im Doppelregister HL die Adresse des laufwerkstypischen DPH (Disk Parameter Header - Disketten Parameter Kopf) übergeben. Ist das gewünschte Laufwerk nicht eingebunden (d.h. ist kein DPH vorhanden), muß HL=0000 übergeben werden.

Beim Aufrufen der Funktion kann unterschieden werden, ob dies ein erster oder ein nachfolgender Aufruf ist. Ist es der erste Aufruf eines Laufwerkes oder wurde (nach einem späteren Aufruf) ein hardware-Fehler gemeldet, so wird die Funktion (mit Bit 0 des Registers E auf 1 gesetzt) aufgerufen, andernfalls ist Bit 0 des Registers auf 0 gesetzt. Ist das Ersterkennungsflag gesetzt (Bit 0=1) muß das Unterprogramm einen Aufruf des Unterprogrammes LOGIN veranlassen. In diesem Unterprogramm kann nun durch verschiedene Mittel festgestellt werden, ob der angegebene DPH zur eingelegten Diskette paßt. (bzgl. der Schreibdichte, der Sektoranzahl oder der Seiten). Ist dies nicht der Fall, kann der DPH angepaßt werden. Nicht verändert werden darf jedoch die Adresse des DPH's.

*** Zum Begriff DPH und LOGIN siehe Kapitel 8 - DISKIO. ***

Dieses Unterprogramm kann in Bank 0 liegen.

```
+-------------------------------------------------------+
! BIOS-Funktion 10  SETTRK (Track setzen)               !
+-------------------------------------------------------+
! Übergabe der Tracknummer zum nächsten Disk-Zugriff    !
+-------------------------------------------------------+
! Eingabe             BC=Tracknummer                    !
! Zurück              nichts                            !
+-------------------------------------------------------+
```

Diese Funktion setzt die Tracknummer wohin der Schreib- Lesekopf vor dem nächsten Diskettenzugriff positioniert werden soll.

Dieses Unterprogramm kann in Bank 0 liegen.

```
+-------------------------------------------------------+
! BIOS_Funktion 11  SETSEC (Sektor setzten)             !
+-------------------------------------------------------+
! Übergabe der Sektornummer zum nächsten Disk-Zugriff   !
+-------------------------------------------------------+
! Eingabe             BC=Sektornummer                   !
! Zurück              Nichts                            !
+-------------------------------------------------------+
```

Diese Funktion setzt die Sektornummer von welchem beim nächsten Diskettenzugriff gelesen oder auf welchen geschrieben werden soll.

Die im Doppelregister BC übergebene Sektornummer ist der Wert, der nach Aufruf der BIOS-Funktion 16 ermittelt wurde (siehe dort).

Dieses Unterprogramm kann auf Bank 0 liegen.

```
+-------------------------------------------------------+
! BIOS-Funktion 12  SETDMA (DMA Adressieren)            !
+-------------------------------------------------------+
! Übergabe der DMA-Adresse                              !
+-------------------------------------------------------+
! Eingabe             BC=DMA-Adresse                    !
! Zurück              nichts                            !
+-------------------------------------------------------+
```

Diese Funktion ermöglicht es die DMA-Adresse (Direct Memory Access - Adresse des direkten Speicherzugriffes) zu setzen.

Ab bzw. an diese Adresse wird der nächste Sektor gelesen bzw geschrieben. Die übergebene DMA-Adresse bleibt erhalten, bis sie von einem weiteren Aufruf geändert wird.

Dieses Unterprogramm kann in Bank 0 liegen.

```
+-----------------------------------------------------------+
! BIOS-Funktion 13  READ ( Lesen )                          !
+-----------------------------------------------------------+
! Lesen eines physikalischen Sektors                        !
+-----------------------------------------------------------+
! Eingabe              keine                                !
! Zurück               A=00  Sektor fehlerfrei gelesen      !
!                      A=01  Sektor nicht lesbar            !
!                      A=FF  Diskette gewechselt ?          !
+-----------------------------------------------------------+
```

Diese Funktion liest den mit SETSEC definierten Sektor von dem mit SETTRK gesetzten Track in den mit SETDMA spezifizierten Speicherbereich von der mit SELDSK angesprochenen Diskette.

Gelesen wird immer ein physikalischer Sektor, d.h. die Sektorgröße die im DPB angegeben wurde (siehe Kapitel 9 DISKIO). Diese Sektorgröße kann bis zu 1024 Bytes groß sein.

Im Gegensatz zu CP/M 2 übernimmt das Sektor -blocking und -deblocking das BDOS. (hierunter ist die 'Umwandlung' der physikalischen Sektorgröße in die logische (128-Byte große) Sektorgröße zu verstehen).

Wurde der Sektor einwandfrei gelesen, wird dem Aufrufer Register A=00 übergeben.

Konnte der Sektor nicht gelesen werden, müssen zunächst mehrere Versuche vorgenommen werden. Im Fehlerfalle muß die Art des Fehlers festgestellt werden.

Stimmen bei der Fehlerfeststellung nur die Diskettenparameter nicht mehr (Density Sektoranzahl Trackanzahl) so muß Register A=FF zurückgegeben werden. In diesem Fall wird vom BDOS noch einmal die Funktion SELDSK aufgerufen, mit Register E Bit 0=1 um ein erneutes LOGIN des Laufwerkes zu erzwingen, da offensichtlich die Diskette gewechselt wurde.

Ist die Diskette mit keinem Mittel zu lesen, liegt vermutlich ein Hardware-Fehler vor (unformatierte Diskette oder defektes Laufwerk). In diesem Falle wird das dem Aufrufer durch Rückgabe von Register A=01H signalisiert.

Dieses Unterprogramm kann in Bank 0 liegen.

BIOS-Funktion 14 WRITE (Schreiben)	
Schreiben eines physikalischen Sektors	
Eingabe	C= deblocking Code
Zurück	A= 00 kein Fehler beim Schreiben
	A= 01 Sektor nicht beschreibbar
	A= 02 Diskette schreibgeschützt
	A= FF Diskette gewechselt ?

Mit dieser Funktion kann ein physikalischer Sektor geschrieben werden.

Es gelten dieselben Bedingungen wie bei der BIOS-Funktion 13. Sollte das Blocking (Aufteilen in 128 Byte Sektoren) vom Benutzer übernommen werden, stehen hierzu in Register C folgende Informationen zur Verfügung:

C=00 Schreiben eines geblockten (späteren) Sektors

C=01 Schreiben eines ungeblockten (nur 128 Byte großen) Sektors.

C=01 Schreiben des ersten Sektors in einem neuen Datenblock

Wie beim Lesen sollte die Funktion im Fehlerfalle mehrere Versuche machen.

Im Fehlerfalle A=FF veranlaßt das BDOS einen erneuten Aufruf von SELDSK um die Diskettenparameter neu einloggen zu können.

Dieses Unterprogramm kann in Bank 0 liegen.

```
+-----------------------------------------------------+
! BIOS-Funktion 15  LISTST  (Druckerstatus)           !
+-----------------------------------------------------+
! Lesen des Drucker(ausgabe)status                    !
+-----------------------------------------------------+
! Eingabe             keine                           !
! Zurück              A=00 Drucker nicht bereit       !
!                     A=FF Drucker bereit             !
+-----------------------------------------------------+
```

Diese Funktion meldet in Register A die Bereitschaft des Druckers.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+-----------------------------------------------------+
! BIOS-Funktion 16  SECTRN (Sektorumsetzung)          !
+-----------------------------------------------------+
! Berechnung der physikalischen Sektornummer          !
+-----------------------------------------------------+
! Eingabe             BC=Logische Sektornummer        !
!                     DE=Adresse der SKEW-Tabelle     !
! Zurück              HL=physikaliche Sektornummer    !
+-----------------------------------------------------+
```

Mit dieser Funktion wird der Sektor-SKEW umgerechnet.

(Siehe hierzu Kapitel 8 DISKIO). Dieser SKEW ist für Disketten im IBM 3740 Format (Standard CP/M 8" einfache Dichte) mit dem Faktor 6 festgelegt.

Die Routine muß die Umrechnung von logischen in physikalische Sektornummern anhand der ebenfalls übergebenen (Adresse der) SKEW-Tabelle berechnen. Wird DE=0000 übergeben, bedeutet dies, daß kein SKEW-Faktor zu berechnen ist, in diesem Falle kann der Inhalt von Register BC direkt in Register HL umgeladen werden.

Es ist zu beachten, daß, entgegen der Gepflogenheiten beim Formatieren, mit Sektor 0 begonnen wird und nicht mit Sektor 1 !

Dieses Unterprogramm kann in Bank 0 liegen.

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
X    Alle nachfolgenden BIOS-Funktionen sind CP/M 3 typisch    X
X    und in der CP/M 2 Sprungleiste nicht enthalten. Es ist    X
X    zu beachten, daß CP/M 2 Systeme auf dem Markt sind,die    X
X    ebenfalls  eine  erweiterte  Sprungleiste haben, diese    X
X    muß aber nicht  notgedrungen  CP/M 3  kompatibel sein.    X
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

```
+--------------------------------------------------------+
! BIOS-Funktion 17  CONOST (Konsolen-Ausgabestatus)       !
+--------------------------------------------------------+
! Festellen ob Konsole ausgabebereit ist                  !
+--------------------------------------------------------+
! Eingabe           keine                                 !
! Zurück            A=00 nicht bereit zur Ausgabe         !
!                   A=FF Konsole bereit zur Ausgabe       !
+--------------------------------------------------------+
```

Mit dieser Funktion kann die Bereitschaft der Konsole, ein weiteres Zeichen auszugeben, getestet werden.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich

```
+--------------------------------------------------------+
! BIOS-Funktion 18  AUXIST (AUX-Eingabestatus)            !
+--------------------------------------------------------+
! Feststellen der Bereitschaft des AUX-Eingabekanals      !
+--------------------------------------------------------+
! Eingabe           keine                                 !
! Zurück            A=00 Nicht bereit zur Eingabe         !
!                   A=FF AUX-Kanal bereit zur Eingabe     !
+--------------------------------------------------------+
```

Diese Funktion testet den AUX-Eingabekanal, ob ein Zeichen eingegeben wurde.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+--------------------------------------------------------+
! BIOS-Funktion 19  AUXOST (AUX-Ausgabestatus)            !
+--------------------------------------------------------+
! Feststellen der Bereitschaft des AUX-Ausgabekanals      !
+--------------------------------------------------------+
! Eingabe           keine                                 !
! Zurück            A=00 wenn nicht bereit                !
!                   A=FF wenn Ausgabekanal bereit         !
+--------------------------------------------------------+
```

Diese Funktion testet den AUX-Kanal, ob ein Zeichen eingegeben wurde.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+----------------------------------------------------------+
! BIOS-Funktion 20  DEVTBL (Vektor der Device-Tabelle) !
+----------------------------------------------------------+
! Zeiger auf DEVICE-Tabelle holen                          !
+----------------------------------------------------------+
! Eingabe            keine                                 !
! Zurück             HL=Zeiger auf DEVICE-Tabelle          !
+----------------------------------------------------------+
```

Durch Aufruf dieser Funktion wird dem 'Rufer' die Startadresse der DEVICE-Tabelle übergeben.

Diese Tabelle (siehe hierzu Kapitel 7 CHARIO) übernimmt die Funktion des IO-Bytes unter CP/M 2. Es sind in der Tabelle alle implementierten (Zeichen)-Schnittstellen zusammengefaßt, so daß über die Tabelle jede Schnittstelle angsprochen werden kann. Der Aufbau eines jeden Eintrages sieht wie folgt aus:

```
db      'DEVNAM'        Name der Schnittstelle z.B. TERM,PRINT usw
                        der Name muss 6 Zeichen lang sein, evtl
                        aufgefuellt mit Leerzeichen

db      00000000b       Beschreibung der Schnittstelle
        !!!!!!!!        wenn entsprechendes BIT=1
        !!!!!!! \______ Eingabeeinheit
        !!!!!! \_______ Ausgabeeinheit
        !!!!! \________ Baudrate durch Software einstellbar
        !!!! \_________ Serienschnittstelle
        !!! \__________ Schnittstelle mit XON/XOFF-Protokoll
         \\\___________ Bit 5..7 nicht belegt

db      baudrate        Angabe der Baudrate nach folgender Tabelle
                        00=keine Baudrate (parallell-betrieb)
                        01=50Baud,02=75 Baud,03=110Baud,04=134,5 Baud
                        05=150Baud,06=300Baud,07=600Baud,08=1200Baud
                        09=1800Baud,0A=2400Baud,0B=3600Baud,0C=4800Baud
                        0D=7200Baud,0E=9600Baud,0F=19200Baud
```

Abgeschlossen wird die Tabelle durch ein NULL-Byte.

Dieses Unterprogramm incl. Tabelle liegt im gemeinsamen Speicherbereich.

```
+---------------------------------------------------------+
! BIOS-Funktion 21  DEVINI (DEV-Initialisierung)          !
+---------------------------------------------------------+
! Initialisierung aller Einheiten in der DEV-Tabelle      !
+---------------------------------------------------------+
! Eingabe            C=DEVice-Nummer (0..15)              !
! Zurück             nichts                               !
+---------------------------------------------------------+
```

Diese Funktion wird üblicherweise nur von dem DEVICE-Befehl angesprochen.

Es kann damit eine Ein- bzw. Ausgabeeinheit spezifisch nachinitialisiert werden. Notwendig ist diese Funktion nur, wenn das XON/ XOFF-Protokoll oder die Baudrate über Software umgeschaltet werden kann.

Sind diese Funktionen nicht erwünscht oder nicht möglich, kann die Funktion mit einem einfachen RETurn abgeschlossen werden.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

```
+---------------------------------------------------------+
! BIOS Funktion 22  DRVTBL (Laufwerkstabelle)             !
+---------------------------------------------------------+
! Zeiger auf Laufwerkstabelle holen                       !
+---------------------------------------------------------+
! Eingabe            keine                                !
! Zurück             HL=Startadresse der Tabelle          !
!                    HL=FFFF wenn keine DRVTBL benutzt    !
!                     Puffer müssen deklariert werden     !
!                     Hashing wird unterstützt            !
!                    HL=FFFE dto. jedoch auch kein        !
!                     Hashing zugelassen (s. GENCPM)      !
+---------------------------------------------------------+
```

Diese Funktion übergibt die Startadresse einer Tabelle mit 16 Vektoren, die auf die 16 möglichen DPH (Disk-Parameter-Header) zeigen. Der erste Eintrag entspricht dabei Laufwerk A, der zweite Eintrag Laufwerk B usw.

Hat ein Laufwerk keinen DPH (ist es also logisch nicht eingebunden), ist der entsprechende Vektor auf 0000 zu setzen.

Soll das Blocking- und Deblocking vom BIOS übernommen werden, (aber wer tut das schon ...) muß HL ebenfalls mit dem Wert FFFE zurückgegeben werden, darüberhinaus muß PSH und PSM im entsprechenden DPB auf NULL gesetzt werden.

Dieses Unterprogramm incl. der Tabelle liegt im gemeinsamen Speicherbereich. Die Vektoren in der Tabelle können jedoch auf Adressen im gebankten Bereich (Bank 0) zeigen, dies bedeutet, daß unter CP/M 3 die Daten des DPH's nur bedingt von TPA-Programmen erreichbar sind.

```
+------------------------------------------------------+
! BIOS-Funktion 23  MULTIO (Mehrfachsektor)            !
+------------------------------------------------------+
! Setzen des Mehrfachsektor-Zählers                    !
+------------------------------------------------------+
! Eingabe             C=Zähler                         !
! Zurück              nichts                           !
+------------------------------------------------------+
```

Diese Funktion hat nur sehr bedingt etwas mit der BDOS-Funktion 44 tun.

Um Speicherinhalte auf oder von Diskette zu transferieren, ruft das BIOS zuerst diese Funktion auf und danach eine Serie von LESE- oder SCHREIB-Befehle. Diese erlaubt dem BIOS (falls entsprechend implementiert) mit einem Zugriff u.U. mehrere Sektoren hintereinander in oder aus dem Puffer zu lesen.

Die maximale Sektorzahl ist abhängig von der physikalischen Sektorgröße, so sind Blöcke von 128X128-Byte Sektoren bis 4X4096-Byte Sektoren mit einem Aufruf möglich. d.h. die maximale Blockgröße entspricht 16K, dem Inhalt eines DIR-Eintrages.

Die Anwendung dieser Funktion wird etwas unübersichtlich, wenn ein Sektor-SKEW verwendet wird; ebenso, wenn unterschiedliche Diskettenformate und Sektorgrößen verwendet werden. Durch die dadurch erheblich aufwendigeren Berechnungsalgorithmen kann der Vorteil eines schnelleren Diskettenzugriffes wieder verloren gehen, daher wird die Funktion relativ selten genutzt.

Dieses Unterprogramm kann im gebankten Speicherbereich liegen.

```
+-----------------------------------------------------------+
! BIOS-Funktion 24  FLUSH (Puffer freigeben)                !
+-----------------------------------------------------------+
! Pufferfreigabe, falls Blocking- und Deblocking vom        !
! BIOS übernommen wird                                      !
+-----------------------------------------------------------+
! Eingabe            keine                                  !
! Zurück             A=00 wenn kein Fehler vorliegt         !
!                    A=01 wenn phys. Fehler vorliegt        !
!                    A=02 wenn Diskette schreibgeschützt!
+-----------------------------------------------------------+
```

Wenn das Blocking- Deblocking (Umwandeln von physikalische in logische Sektoren oder umgekehrt) vom BIOS übernommen wird, müssen nach Aufruf dieser Funktion noch nicht geschriebene (logische) Sektoren aus dem Puffer auf Diskette geschrieben werden.

Darunter ist folgendes zu verstehen:

Die physikalische Sektorgröße sei 1024 Bytes pro Sektor, dies entspricht 8 logischen Sektoren zu je 128 Bytes. Soll nun z.B. ein Programm von 1500 Bytes (knapp 12 logische Sektoren) auf Diskette geschrieben werden, so werden zuerst 8 logische Sektoren zu einem 1024 Byte-Block zusammengefaßt. Dieser Block wird dann auf Diskette geschrieben. Danach werden die nächsten 4 logischen Sektoren in den Puffer geschrieben, ein physikalischer Zugriff würde jedoch erst erfolgen, wenn die Sektorgröße von 1024 Bytes erreicht wäre. Ist das BDOS jedoch mit dem Datentransfer zu Ende, ruft es zum Abschluß die Funktion FLUSH auf, nun muß der letzte Pufferinhalt, sei der Puffer voll oder nicht, auf Diskette geschrieben werden, um die gesamte Datei zu sichern.

Wird diese Funktion nicht benötigt, so muß sie beim Aufruf das Register A=00 zurückgeben.

Dieses Unterprogramm kann in Bank 0 liegen

```
+--------------------------------------------------------+
! BIOS-Funktion 25  MOVE (Speichertransfer)              !
+--------------------------------------------------------+
! Transfer von Speicherinhalten                          !
+--------------------------------------------------------+
! Eingabe              HL=Zieladresse des Transfer       !
!                      DE=Quelladresse des Transfer      !
!                      BC=Länge des Speicherblockes      !
! Zurück               HL und DE müssen auf die jeweils  !
!                      letzte Adresse +1 des Transfers   !
!                      zeigen.                           !
+--------------------------------------------------------+
```

Diese Funktion übernimmt den Datentransfer innerhalb einer Bank oder auch zwischen verschiedenen Banken, wenn zuvor die BIOS-Funktion 29 aufgerufen wurde.

Es ist zu beachten, daß die Quell- und Zielzeiger zur Anwendung des Z80-Befehles LDIR in umgekehrter Reihenfolge stehen.

Diese Funktion liegt im gemeinsamen Speicherbereich.

```
+------------------------------------------------------+
! BIOS-Funktion 26  TIME (Datum und Uhrzeit)           !
+------------------------------------------------------+
! Setzen oder Lesen von Uhrzeit und Datum              !
+------------------------------------------------------+
! Eingabe            C=00 Uhrzeit und Datum aus der Uhr !
!                         lesen und in den SCB schreiben!
!                    C=FF Uhrzeit und Datum aus dem SCB !
!                         lesen und Uhr entsprechend    !
!                         nachstellen                   !
+------------------------------------------------------+
```

Diese Funktion erlaubt die Implementation von Datum und Uhrzeit in einem CP/M 3 System.

Ist die Uhrzeit interrupt-getrieben, muß das Einschreiben der Uhrzeit in den SCB von der entsprechenden service-Routine übernommen werden. Bei einem Aufruf mit C=00 kann die Funktion dann mit einem einfachen RETurn abgeschlossen werden.

Das Format, unter welchem die Uhrzeit und das Datum abgelegt wird, ist unter der BDOS-Funktion 104 beschrieben.

Wird TIME nicht unterstützt, so kann die Funktion mit REturn abgeschlossen werden. Das BDOS wird trotzdem das Datum und die Uhrzeit im SCB setzen, von wo aus es für Benutzerprogramme zugänglich ist, es bleibt dabei lediglich die Uhr 'stehen'.

Dieses Unterprogramm liegt im gemeinsamen Bereich, Teile davon können im gebankten Bereich (Bank 0) liegen, die Umschaltung wird jedoch intern vorgenommen, so daß TPA-Programme auf die Funktion zugreifen können. Die Benutzung der BDOS-Funktionen 104..105, um die Uhrzeit zu 'erfragen' ist jedoch meist sinnvoller.

BIOS-Funktion 27 SELMEM (Speicherbank einstellen)	
Einstellen einer Speicherbank	
Eingabe	A=Speicherbank 00=0,01=1 usw
Zurück	nichts

Mit dieser Funktion wird die Bank umgeschaltet.

Es können 16 Banken zu je 64k genutzt werden. Alle Banken müssen unter CP/M 3 einen gemeinsamen Speicherbereich von 4..8K von Adresse FFFF in Richtung Adresse 0000 haben. Der gemeinsame Bereich muß in allen Banken gleich groß sein.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich, kann aus der TPA heraus jedoch nicht aufgerufen werden, da die Rückkehradresse ja durch die Bankumschaltung nicht mehr stimmt.

BIOS-Funktion 28 SETBNK (DMA-Bank einstellen)	
Einstellen der Speicherbank für den nächsten DMA-Datentransfer	
Eingabe	A=Speicherbank
Zurück	nichts

Diese Funktion entspricht der Funktion 27, hier wird jedoch die Bank festgelegt, in oder aus welcher der nächste LESE- oder SCHREIB-Befehl die Daten transferiert.

Dieses Unterprogramm liegt im gemeinsamen Speicherbereich.

BIOS-Funktion 29 XMOVE (erweiterter Datentransfer)	
Transfer von Daten zwischen den Banken	
Eingabe	B=Zielbank (wohin) C=Quellbank (woher)
Zurück	nichts

Diese Funktion ermöglicht den Datentransfer zwischen den Banken.

Sie wird zu dem alleinigen Zweck aufgerufen, die Adressen der Ziel- und Quellbank abzulegen und ein Flag zu setzen.

Der eigentliche Transfer geschieht erst mit Aufruf der BIOS-Funktion 25 (MOVE), die das entsprechende XMOVE-Flag bei jedem Aufruf prüfen und den Datentransfer entsprechend handhaben muß.

Diese Funktion liegt im gemeinsamen Speicherbereich.

```
+-------------------------------------------------------+
! BIOS Funktion 30..32      Reserve                     !
+-------------------------------------------------------+
```

Diese drei Einsprünge sind von Digital Research als Reserve deklariert.

Funktion 30 ist jedoch für spezielle System-Implementationen freigegeben, kann vom Benutzer also frei verwendet werden.

Dieser Einsprung wird zwar vom BDOS nicht verwendet, bietet sich jedoch an, spezielle Hardware-Treiber einzubinden, die von TPA-Programmen aufgerufen werden können. Sind mehrere Treiber eingebunden, kann die Funktion z.B. die Adresse einer weiteren Funktionstabelle übergeben.

Kapitel 6 Das BIOS-Kernprogramm und der SCB

Der Übersichtlichkeit halber ist das BIOS normalerweise in verschiedene Funktionsblöcke aufgespaltet, die einzeln einfacher zu überblicken und zu bearbeiten sind.

Zwei Programmsegmente sind jedoch mehr oder weniger unveränderbar:

1. Die SCB-Vektortabelle
2. Das BIOS-Kernprogramm

Der SCB (System Kontroll Block) besteht im Prinzip nur aus Adressreferenzen, auf welche die verschiedenen BIOS-Programmteile als EXTERNAL (externe Linkadresse) zugreifen. Die im SCB genannte BASIS-Adresse ist rein fiktiv. Sie wird ebenfalls wie alle Offset-Adressen vom Generierungsprogramm GENCPM entsprechend den Systemgegebenheiten angepaßt. Dies darf jedoch unter keinen Umständen dazu führen, daß im BIOS Adressen in dem angegeben Bereich für andere Dinge als dem SCB-Zugriff benutzt werden.

Der SCB-Bereich ist ein Teil des CP/M 3 BDOS, der Vektoren und Daten aus den verschiedensten Bereichen enthält. Alle CP/M 3 Programmteile wie CCP, BDOS und BIOS können auf den SCB direkt zugreifen. Über bestimmte BDOS-Aufrufe kann auch aus TPA-Programmen auf die entsprechenden Daten zugegriffen werden.

Nicht alle Daten des SCB's sind von Digital Research bekanntgemacht worden, die Adressen für das BIOS sind im nachfolgenden Listing zusammengefaßt:

```
;*****************************************************************
;*                                                               *
;* rel 1.0 vom 07 01.85                                          *
;*                                                               *
;* unveraenderbarer Teil des BIOSKRNL.ASM                        *
;* CP/M greift auf die Vektoren in dieser Tabelle zu um          *
;* mit dem BIOS korrespondieren zu koennen                       *
;* Zu beachten ist, die Tabelle ist NICHT WIRKLICH unter         *
;* SCB$BASE zu finden - CP/M transferiert sie an andere          *
;* Stelle - wohin ???                                            *
;* Zugriff aus Anwenderprogramme auf SCB$BASE ist daher          *
;* sinnlos. Im BIOS kann jedoch ohne weiteres auf die            *
;* Labels zugegriffen werden.                                    *
;*                                                               *
;* geschrieben von RAOUL O.KOERBER                               *
;* nach Unterlagen von DRI                                       *
;*                                                               *
;*****************************************************************

        ; ****************
        ; Systemadressen
        ; ****************

        public @civec, @covec, @aivec, @aovec, @lovec, @bnkbf
        public @crdma, @crdsk, @vinfo, @resel, @fx, @usrcd
        public @mltio, @ermde, @erdsk, @media, @bflgs
        public @date, @hour, @min, @sec, ?erjmp, @mxtpa
```

```
                    ; ****************
                    ; SCB-Basis Adresse
                    ; ****************

FE00    scb$base equ    0FE00H                  ; Basis des SCB

                    ; ****************
                    ; DEVICE-Auswahl              ; Character+Disk
                    ; ****************

                    ; in jedem Byte kann gesetzt werden, welche Einheit
                    ; angesprochen werden kann. BIT 15 entspricht Einheit 0
                    ; BIT 4 entspricht Einheit 11. Die BITs 3..0 sind von
                    ; DRI fuer spaetere Erweiterungen reserviert.
                    ; Die Auswahl kann direkt ueber das CP/M Programm
                    ; DEVICE.COM erfolgen. Der Zugriff erfolgt auf die
                    ; Namen im Programmteil CHARIO

FE22    @CIVEC  equ     scb$base+22h            ; KONSOLE Eingabe
FE24    @COVEC  equ     scb$base+24h            ; KONSOLE Ausgabe
FE26    @AIVEC  equ     scb$base+26h            ; AUX Eingabe
FE28    @AOVEC  equ     scb$base+28h            ; AUX Ausgabe
FE2A    @LOVEC  equ     scb$base+2Ah            ; LIST Ausgabe

                    ; ****************
                    ; BANK-Puffer                 ; im gebankten System
                    ; ****************

FE35    @BNKBF  equ     scb$base+35h            ;( banked BIOS)

                    ; ****************
                    ; CP/M Variable               ; fuer BDOS
                    ; ****************

FE3C    @CRDMA  equ     scb$base+3Ch            ; Derzeitige DMA Adresse
FE3E    @CRDSK  equ     scb$base+3Eh            ; Angesprochenes Laufwerk
FE3F    @VINFO  equ     scb$base+3Fh            ; BDOS Variable "INFO"
FE41    @RESEL  equ     scb$base+41h            ; FCB Flag
FE43    @FX     equ     scb$base+43h            ; BDOS Funktion fuer Fehlermeldungen
FE44    @USRCD  equ     scb$base+44h            ; Derzeitige USER-Nummer
FE4A    @MLTIO  equ     scb$base+4Ah            ; Derzeitiger Multi-Sector Zaehler
FE4B    @ERMDE  equ     scb$base+4Bh            ; BDOS Error Mode
FE51    @ERDSK  equ     scb$base+51h            ; BDOS Error Disk
FE54    @MEDIA  equ     scb$base+54h            ; Gesetzt vom BIOS bei 'open door'
FE57    @BFLGS  equ     scb$base+57h            ; BDOS Message Size Flag
```

```
                        ; ****************
                        ; Datum und Uhrzeit              ; Variable hier ablegen
                        ; ****************

FE58       @DATE   equ     scb$base+58h            ; Datum in Tagen seit dem 1 Jan 78 !
FE5A       @HOUR   equ     scb$base+5Ah            ; Stunde in BCD (Byte, r/w)
FE5B       @MIN    equ     scb$base+5Bh            ; Minute in BCD (Byte, r/w)
FE5C       @SEC    equ     scb$base+5Ch            ; Sekunde in BCD (Byte, r/w)

                   ; ****************
                   ; Aussprung Fehlermeldung
                   ; ****************

FE5F       ?ERJMP  equ     scb$base+5Fh            ; BDOS Error Message Jump

                   ; ****************
                   ; Startadresse BDOS
                   ; ****************

FE62       @MXTPA  equ     scb$base+62h            ; Ende der TPA ( Start des BDOS )

                   end
```

Das BIOS-Kernprogramm, (BIOSKRNL.Z80) ebenfalls (relativ) Hardware-unabhängig, ist der erste Programmteil des BIOS.

Hier befindet sich, gleich am Anfang des Programmteiles, die BIOS-Sprungtabelle. Im BIOS-Kernprogramm werden die Sprünge in die entsprechenden anderen Programmsegmente weitergeleitet oder noch zusätzliche Daten aufbereitet. Im nachfolgenden Listing sind alle Routinen ausreichend kommentiert.

Beim Kaltstart eines CP/M 3 Systems wird aus dem CPM-Lader (CPMLDR) heraus, nachdem die Datei CPM3.SYS (das eigentliche Betriebssystem) geladen wurde, nach Adresse ?BOOT gesprungen und damit die letzte Systeminitialisierung vorgenommen.

Es ist zu berücksichtigen, daß vom CPM-Lader heraus noch keinerlei Bankumschaltung vorgenommen wurde, das Laderprogramm wurde vom BOOT-Monitor jedoch (u.U. mit Hilfe des Track-0-Laders) nach Bank 0 geladen.

Ohne zwingenden Grund sollte werde die Datei SCB.Z80 noch die Datei BIOSKRNL.Z80 geändert werden - je weniger Änderungen gemacht werden - je größser ist die Kompatibilität zu 'anderen' Systemen.

```
        MACLIB  DEFAULT.INC      ; Einlesen Vereinbarungen
        MACLIB  MODEBAUO.INC
;**********************************************************************
;*                                                                    *
;* rel 1.0 vom 07.01.85                                               *
;*                                                                    *
;* Dies ist das Hauptmodul eines BIOS fuer CP/M3                      *
;* mit Systemkarten aus dem NDR-Computer Programm                     *
;* Dieser Programmteil ist invariabel und sollte                      *
;* moeglichst so belassen werden.                                     *
;*                                                                    *
;*                                                                    *
;* nach Unterlagen von DRI                                            *
;*                                                                    *
;**********************************************************************

        ;
        ; ****************
        ; Systemadressen
        ; ****************
        ;


        ;
        ; Variable im gemeinsamen RAM-Fenster
        ;

        extrn   @covec,@civec,@aovec,@aivec,@iovec      ; I/O Vektoren
        extrn   @mxtpa                  ; System-Einsprung-Vektor
        extrn   @bnkbf                  ; 128 byte scratch-Puffer

        ;
        ; Initialisierung
        ;

        extrn   ?init                   ; Hauptinitialisierung und HALLO
        extrn   ?ldccp,?rlccp           ; (Nach)-Laden des CCP bei (W)-BOOT

        ;
        ; Benutzerdefinierte Zeichen-Ein/Ausgabe Routinen
        ;

        extrn   ?ci,?co,?cist,?cost     ; Aufruf mit DEVICE-Nummer in <B> !
        extrn   ?cinit                  ; (RE)-Init mit DEVICE-Nummer in <C>
        extrn   @ctbl                   ; DEVICE-Tabelle

        ;
        ; Vektoren zum Disketten-Zugriff
        ;

        extrn   motoff                  ; Laufwerksabschaltung
        extrn   @dtbl                   ; Zeiger auf XDPHs
        public  @drv,@trk,@sect         ; Parameter fuer Disketten I/O
        public  @dma,@dbnk,@cnt         ; dto

        ;
        ; Speicher-Kontrolle ( banking )
        ;

```

```
                                public  @cbnk                   ; Vektor in gesetzte ( current )-Bank
                                extrn   ?xmove,?move            ; Interbank move ( Transfer )
                                extrn   ?bank                   ; Auswahl (CPU)-Bank

                                ;
                                ; Uhrzeit und Datum
                                ;

                                extrn   ?time

                                ;
                                ; Sonstige benutzbare Unterprogramme
                                ;

                                public  ?pmsg,?pdec             ; 'print message'- 'print decimal'
                                public  ?pderr                  ; 'print FEHLER'
                                public  ipchl                   ; indirekter CALL

                                ;
                                ; Extern aufrufbare Einspruenge in die BIOS Einsprungtabelle
                                ;

                                public  ?boot,?wboot,?const,?conin,?cono,?list,?auxo,?auxi
                                public  ?home,?sldsk,?sttrk,?stsec,?stdma,?read,?write
                                public  ?lists,?sctrn
                                public  ?conos,?auxis,?auxos,?dvtbl,?devin,?drtbl
                                public  ?mltio,?flush,?mov,?tim,?bnksl,?stbnk,?xmov

                                cseg                            ; Residenter RAM-Teil

                                ;
                                ;*************************************************************
                                ;*                                                           *
                                ;* BIOS Sprungtabelle                                        *
                                ;*                                                           *
                                ;* alle BIOS-Routinen werden vom BDOS ueber diese Sprungtabelle *
                                ;* aufgerufen. Bitte beachten, dass einige Vektoren in den   *
                                ;* residenten RAM-Teil zeigen ( E000...FFFF) andere dagegen  *
                                ;* auf Bank 0 ! ( mit ** gekennzeichnet ) auf diese Bank     *
                                ;* wird beim kaltstart zugegriffen.                          *
                                ;*                                                           *
                                ;*************************************************************
                                ;

0000' C3 0000"      ?boot:  jp      boot                    ; ** Kaltstart
0003' C3 006C'      ?wboot: jp      wboot                   ; Warmstart
0006' C3 0152'      ?const: jp      const                   ; Konsole Eingabestatus
0009' C3 016C'      ?conin: jp      conin                   ; Konsole Eingabe
000C' C3 00BA'      ?cono:  jp      conout                  ; Konsole Ausgabe

000F' C3 00C4'      ?list:  jp      list                    ; Drucker Ausgabe
0012' C3 00BF'      ?auxo:  jp      auxout                  ; AUX Ausgabe
0015' C3 0174'      ?auxi:  jp      auxin                   ; AUX Eingabe
0018' C3 0056"      ?home:  jp      .home                   ; ** Diskette auf Track 0 setzen
001B' C3 0034"      ?sldsk: jp      seldsk                  ; ** Auswahl Laufwerk

001E' C3 0059"      ?sttrk: jp      settrk                  ; ** Track setzen
0021' C3 005E"      ?stsec: jp      setsec                  ; ** Sektor setzen
```

```
0024' C3 0063"   ?stdma: jp     setdma               ; ** Disketten I/O-RAM setzen
0027' C3 0079"   ?read:  jp     read                 ; ** Sektor lesen
002A' C3 0081"   ?write: jp     write                ; ** Sektor schreiben

002D' C3 00EC'   ?lists: jp     listst               ; Drucker Status
0030' C3 006E"   ?sctrn: jp     sectrn               ; ** log = phys Sektor Berechnung
0033' C3 00E2'   ?conos: jp     conost               ; Konsole Ausgabestatus
0036' C3 0157'   ?auxis: jp     auxist               ; AUX Eingabestatus
0039' C3 00E7'   ?auxos: jp     auxost               ; AUX Ausgabestatus

003C' C3 00B2'   ?dvtbl: jp     devtbl               ; holt Adresse von @CTBL
003F' C3 0000#   ?devin: jp     ?cinit               ; Baudratewechsel
0042' C3 00B6'   ?drtbl: jp     getdrv               ; holt Adresse von @DTBL
0045' C3 00A1"   ?mltio: jp     multio               ; ** DD mehrfach Disk I/O pro Sektor
0048' C3 00A5"   ?flush: jp     flush                ; ** Disk flush

004B' C3 0000#   ?mov:   jp     ?move                ; Block move (RAM)
004E' C3 0000#   ?tim:   jp     ?time                ; ** Zeit und Datum
0051' C3 01ED'   ?bnksl: jp     bnksel               ; Bank fuer DMA und Programm
0054' C3 006A"   ?stbnk: jp     setbnk               ; Bank fuer DMA und Bank fuer Programm
0057' C3 0000#   ?xmov:  jp     ?xmove               ; extended INTER-Bank move

005A' C3 0000            jp     0                    ; Reserviert fuer
005D' C3 0000            jp     0                    ; spaetere Ergaenzungen
0060' C3 0000            jp     0                    ; von DRI


                         ;
                         ;*****************************************************************
                         ;*                                                               *
                         ;* Beginn des Programmteiles                                     *
                         ;*                                                               *
                         ;* Das Programm wurde in CSEG Segmenten ( residenter Bereich)    *
                         ;* und in DSEG Segmenten ( gebankter Bereich ) aufgeteilt. Die   *
                         ;* Aufteilung ( wohin was ) uebernimmt das CP/M Programm         *
                         ;* GENCPM.COM .                                                  *
                         ;*                                                               *
                         ;*****************************************************************
                         ;

                         dseg                        ; Bank 0

                         ;
                         ; ********************************
                         ; hardeware-Initialisierung(en)
                         ; ********************************
                         ;
                         ; ***************
                         ; Kaltstart
                         ; ***************
                         ;

0000"            boot:


                         ;
                         ; Kaltstart des Systems nachdem es von CPMLDR
                         ; geladen wurde. Es werden keine Daten uebergeben
                         ;

```

```
0000" 31 0082'          ld      sp,boot$stack           ; Stack initialisieren
                        ;
                        ; ****************
                        ; hardware-Init
                        ; ****************
                        ;
                        ; Initialisierung aller Einheiten, die nicht bereits
                        ; vom Monitor initialisiert wurden, AUSSER Disk- und
                        ; Zeichen ( Charactr ) I/O
                        ; z.B. soft/hardware-Uhr
                        ;
0003" CD 0000#          call    ?init                   ; hard init
                        ;
                        ; ****************
                        ; E/A-Init                      ; Nur Zeichen Ein/Ausgabe
                        ; ****************
                        ;
0006" 0E 0F             ld      c,15                    ; INIT aller 16 moeglichen DEVICES
0008"               c$init$loop:
0008" C5                push    bc                      ; retten von C, dem Devicezaehler
0009" CD 0000#          call    ?cinit                  ; einzelne Devices initialisieren
000C" C1                pop     bc                      ; Devicezaehler
000D" 0D                dec     c                       ; -1
000E" F2 0008"          jp      p,c$init$loop           ; Schleife bis fertig
                        ;
                        ; ****************
                        ; DISK-Init                     ; Initialisierung von FDC's
                        ; ****************
                        ;
                        ; Initialisierung  aller (implementierter) Laufwerke
                        ; ( siehe auch dort ) - Welches Laufwerk als implementiert
                        ; betrachtet wird steht im Programmteil DRVTBL.
                        ; Nur was dort eingetragen ist gilt als resident.
0011" 06 10             ld      b,16                    ; 16 Laufwerke (moeglich)
0013" 21 0000#          ld      hl,@dtbl                ; Zeiger auf XDPH-Vektoren
0016"               d$init$loop:                        ; Initschleife
0016" C5                push    bc                      ; Schleifenzaehler retten
0017" 5E                ld      e,(hl)                  ; Zeiger auf XDPH einlesen
0018" 23                inc     hl                      ; nach DE
0019" 56                ld      d,(hl)                  ;
001A" 23                inc     hl
001B" 7B                ld      a,e                     ; fertig wenn DE=0
001C" B2                or      d                       ; => kein weiteres Laufwerk
001D" 28 0F             jr      z,d$init$next
001F" E5                push    hl                      ; rette Zeiger auf @dtbl
0020" E5                push    hl                      ; gleich 2*
0021" EB                ex      de,hl                   ; HL= Adresse XDPH
0022" 2B                dec     hl                      ; HL zeigt auf TYPE
```

```
0023" 2B                dec     hl                      ; HL zeigt auf UNIT
0024" 2B                dec     hl                      ; HL zeigt auf HIGH Init
0025" 56                ld      d,(hl)
0026" 2B                dec     hl                      ; HL zeigt auf LOW Init
0027" 5E                ld      e,(hl)                  ; Adresse der LW-Init in DE
0028" EB                ex      de,hl                   ; HL=INIT
0029" D1                pop     de                      ; DE=Zeiger auf XDPH
002A" CD 008A"          call    ipchl                   ; INIT des Laufwerkes ausfuehren
002D" E1                pop     hl                      ; Original Zeiger in @dtbl

002E"           d$init$next:                            ; Schleife

002E" C1                pop     bc                      ; Zaehler
002F" 10 E5             djnz    d$init$loop             ; Schleife pro Laufwerk
0031" C3 0063'          jp      boot$1                  ; weiter im residenten RAM-Teil

                        cseg                            ; residenter RAM-Teil

0063'           boot$1:

                ;
                ; ********************************
                ; System-software-Initialisierung
                ; ********************************
                ;

0063' CD 0078'          call    set$jumps               ; JUMP bei 0 und 5 Init

                ;
                ; ab hier haben wir auf die TPA-Bank (Bank 1) geschaltet !
                ;

0066' CD 0000#          call    ?ldccp                  ; ... und CCP laden
0069' C3 0100           jp      ccp                     ; exit nach CCP

006C'           wboot:

                ;
                ; ****************
                ; Warmstart
                ; ****************
                ;

006C' 31 0062'          ld      sp,boot$stack           ; Stack auch bei Warmstart
006F' CD 0078'          call    set$jumps               ; JUMP bei 0 und 5 setzen

                ;
                ; ab hier haben wir auf Bank 1 geschaltet
                ;

0072' CD 0000#          call    ?rlccp                  ; CCP nachladen
0075' C3 0100           jp      ccp                     ; ... und ausfuehren ...

                ;
                ; ****************
                ; Hilfsprogramme
                ; ****************
                ;
```

```

0078'              set$jumps:

                             ;
                             ; init der JUMP-Info bei Adresse 0 und 5 (CP/M Einsprung)
                             ; in gebankten Systemen wird jetzt auf die TPA-Bank geschaltet
                             ;

             FFFF       if   banked                          ; wenn gebankt

0078' 3E 01                  ld      a,1                     ; Bank 1 Laden ( TPA )
007A' CD 0051'               call    ?bnksl

                        endif

007D' 3E C3                  ld      a,0C3H                  ; JUMP Instruction
007F' 32 0000                ld      (0),a                   ; nach 0 und
0082' 32 0005                ld      (5),a                   ; nach 5 laden
0085' 21 0003'               ld      hl,?wboot               ; Warmstart ueber CBIOS
0088' 22 0001                ld      (1),hl                  ; nach 1 ( 0=JP ?WBOOT )
008B' 2A 0000#               ld      hl,(@mxtpa)             ; Einsprung in's BDOS steht dort
008E' 22 0006                ld      (6),hl                  ; nach 6 ( 5=JP BDOS )
0091' C9                     ret


                             ;
                             ; ***************
                             ; Bootstack ...
                             ; ***************
                             ;

0092'        0020            defs    32                      ; 16 * push

             00B2'     boot$stack    equ     $               ; 16 * duerfte genuegen


                             ;
                             ; ******************************
                             ; Adresse der DEVICE-Tabellen lesen
                             ; ******************************
                             ;

00B2' 21 0000#         devtbl: ld    hl,@ctbl                ; Startadresse der
00B5' C9                     ret                             ; CHARACTER DEVICE Tabelle

00B6' 21 0000#         getdrv: ld    hl,@dtbl                ; Startadresse der
00B9' C9                     ret                             ; DRIVE-Tabelle


                             ;
                             ; ******************************
                             ; logische Zeichen-Ausgabe
                             ; ******************************
                             ;
                             ; Der physikalische Zugriff erfolgt  erst in CHARIO.ASM
                             ;


                             ;
                             ; ***************
                             ; Konsole
                             ; ***************
```

```
                                ;
00BA'                   conout:
                                ;
                                ; Ausgabe des Zeichens in <C> an ALLE angewaehlten
                                ; Konsolen-Ausgabeeinheiten
                                ;
00BA' 2A 0000#                  ld      hl,(@covec)             ; Zeiger auf Ausgabe Bit-Vektor
00BD' 18 08                     jr      out$scan
                                ;
                                ; ****************
                                ; AUX
                                ; ****************
                                ;
00BF'                   auxout:
                                ;
                                ; Ausgabe des Zeichens in <C> an ALLE angewaehlten
                                ; AUX-Ausgabeeinheiten (was auch immer das ist)
                                ;
00BF' 2A 0000#                  ld      hl,(@aovec)             ; Zeiger auf AUX Ausgabe Bit-Vektor
00C2' 18 03                     jr      out$scan
                                ;
                                ; ****************
                                ; Drucker
                                ; ****************
                                ;
00C4'                   list:
                                ;
                                ; Ausgabe des Zeichens in <C> an ALLE angewaehlten
                                ; LIST-Ausgabeeinheiten (Drucker)
                                ;
00C4' 2A 0000#                  ld      hl,(@lovec)             ; Zeiger auf LIST Ausgabe Bit-Vektor
00C7'                   out$scan:
                                ;
                                ; Schleife zur Abfrage welche Einheit bereit zur Ausgabe ist
                                ;  Aktiviert ist die Einheit fuer die ein BIT im entsprechenden
                                ; SCB-Vektor gesetzt ist. Bit7 entspricht dabei DEVICE 0 und Bit4 ist
                                ; DEVICE 11. Bit 0..3 ist von CP/M reserviert
                                ; Das Setzen der entsprechenden Startbedingungen wird in ?INIT
                                ; vorgenommen ansonsten von DEVICE.COM
                                ; Auszugebendes Zeichen in <C> wird in <A> zurueckerwartet
                                ;
00C7' 06 00                     ld      b,0                     ; Start mit Einheit 0
00C9'                   co$next:                                ; Schleife
```

```
00C9' 29                  add     hl,hl                ; nach links schieben MSB in CARRY
00CA' 30 0F               jr      nc,not$out$device    ; wenn DEVICE nicht angesprochen ...
00CC' E5                  push    hl                   ; sonst BIT-Zeiger retten
00CD' C5                  push    bc                   ; und Einheit-Nummer ( in <B> )

00CE'           not$out$ready:

00CE' CD 0103'            call    coster               ; warte bis Einheit bereit
00D1' B7                  or      a                    ; FLAG (0 = (noch) NICHT bereit)
00D2' 28 FA               jr      z,not$out$ready      ; Schleife bis bereit ...
00D4' C1                  pop     bc                   ; Einheit-Nummer zurueck und erneut
00D5' C5                  push    bc                   ; retten. Auszugebendes Zeichen in <C>
00D6' CD 0000#            call    ?co                  ; Ausgabe auf Device, Nummer in <B>
00D9' C1                  pop     bc                   ; Einheit-Nummer und Zeichen
00DA' E1                  pop     hl                   ; BIT-Zeiger

00DB'           not$out$device:

00DB' 04                  inc     b                    ; naechste Einheit
00DC' 7C                  ld      a,h                  ; Fertig? - dann <HL>=0 !
00DD' B5                  or      l
00DE' 20 E9               jr      nz,co$next           ; ... sonst weiter
00E0' 79                  ld      a,c                  ; Zeichen zurueck in <A>
00E1' C9                  ret

                          ;
                          ; *******************************
                          ; Ausgabestatus
                          ; *******************************
                          ;

                          ;
                          ; ****************
                          ; Konsole
                          ; ****************
                          ;

00E2'           conost:

                          ;
                          ; Konsolen-Ausgabe- Status, <A>=FF wenn alle angewaehlten Ausgabe-
                          ; Einheiten bereit sind, sonst <A>=00
                          ;

00E2' 2A 0000#            ld      hl,(@covec)          ; Zeiger auf Konsolen BIT-Vektor
00E5' 18 08               jr      ost$scan

                          ;
                          ; ****************
                          ; AUX
                          ; ****************
                          ;

00E7'           auxost:

                          ;
                          ; AUX-Ausgabe-Status ... wie CONOST fuer AUX
```

```
                               ;
 00E7' 2A 0000#                ld      hl,(@aovec)             ; Zeiger auf AUX BIT-Vektor
 00EA' 18 03                   jr      ost$scan
                               ;
                               ; ****************
                               ; Drucker
                               ; ****************
                               ;
 00EC'                 listst:
                               ;
                               ; LIST-Ausgabe-Status ... wie CONOST fuer LIST
                               ;
 00EC' 2A 0000#                ld      hl, (@lovec)            ; Zeiger auf LIST BIT-Vektor
 00EF'                 ost$scan:
                               ;
                               ; Schleife zur Abfrage welche Einheit fertig ist
                               ; ( siehe hierzu auch OUT$SCAN )
                               ;
 00EF' 06 00                   ld      b,0                     ; Start mit Einheit 0
 00F1'                 cos$next:
 00F1' 29                      add     hl,hl                   ; MSB nach CARRY
 00F2' E5                      push    hl                      ; BIT-Vektor retten
 00F3' C5                      push    bc                      ; Zaehler auch
 00F4' DC 0103'                call    c,coster                ; pruefen ob fertig wenn DEV selektiert
 00F7' C1                      pop     bc                      ; Zaehler ...
 00F8' E1                      pop     hl                      ; BIT-Vektor
 00F9' B7                      or      a                       ; ... war DEVICE fertig ?
 00FA' C8                      ret     z                       ; NEIN ... zurueck mit 0 ( falsch )
 00FB' 04                      inc     b                       ; sonst DEVICE-Nummer +1
 00FC' 7C                      ld      a,h                     ; pruefen ob fertig
 00FD' B5                      or      l
 00FE' 20 F1                   jr      nz,cos$next             ; Schleife bis alle DEV abgefragt
 0100' F6 FF                   or      -1                      ; alle selektierten waren bereit
 0102' C9                      ret                             ; also A=FF ( true )
                               ;
                               ; ****************
                               ; Hilfsprogramme
                               ; ****************
                               ;
 0103'                 coster:
                               ;
                               ; pruefe Ausgabeeinheit ob BEREIT ( incl optionalem XON/XOFF )
                               ; erwartet die Device-nummer in <B>
                               ;
```

```
0103' 68              ld      l,b             ; DEVICE Nummer
0104' 26 00           ld      h,0             ; als Offset in Tabelle
0106' E5              push    hl              ; Offset retten
0107' 29              add     hl,hl           ; *2
0108' 29              add     hl,hl           ; *4
0109' 29              add     hl,hl           ; *8
010A' 11 0006#        ld      de,@ctbl+6      ; Tabelle ( nach Device-Namen )
010D' 19              add     hl,de           ; Offset
010E' 7E              ld      a,(hl)          ; lese TYP
010F' E6 10           and     mb$xon$xoff     ; XON-XOFF Protokoll ?
0111' E1              pop     hl              ; Offset
0112' CA 0000#        jp      z,?cost         ; kein XON/XOFF-Status direkt holen

                      ;
                      ; lesen mit XON/XOFF Protokoll
                      ;

0115' 11 01F3'        ld      de,xoflist
0118' 19              add     hl,de           ; Zeiger auf richtigen XON/XOFF-Flag
0119' CD 013B'        call    cistl           ; Status lesen
011C' 7E              ld      a,(hl)          ; XON/XOFF Flag
011D' C4 0149'        call    nz,ci1          ; hole FLAG oder lese Eingabe
0120' FE 11           cp      ctlq
0122' 20 02           jr      nz,not$q        ; wenn CNTRL-Q,
0124' 3E FF           ld      a,-1            ; dann READY-Flag setzen

0126' FE 13   not$q:  cp      ctls
0128' 20 01           jr      nz,not$s        ; wenn CNTRL-S
012A' AF              xor     a               ; READY-Flag loeschen
012B' 77      not$s:  ld      (hl),a          ; Flag retten
012C' CD 0142'        call    costl           ; aktuellen Ausgabestatus lesen
012F' A6              and     (hl)            ; maskieren mit CNTRL Q/S Flag
0130' C9              ret

0131'         col:

                      ;
                      ; Ausgabe ueber <A> mit <DE>,<BC> und <HL> gerettet
                      ;

0131' C5              push    bc
0132' E5              push    hl
0133' D5              push    de
0134' 4F              ld      c,a
0135' CD 000C'        call    ?cono
0138' D1              pop     de
0139' 18 13           jr      ci2

013B'         cistl:

                      ;
                      ; Eingabe-Status mit <BC> und <HL> gerettet
                      ;

013B' C5              push    bc
013C' E5              push    hl
013D' CD 0000#        call    ?cist
0140' 18 0C           jr      ci2
```

```
0142'                costl:
                             ;
                             ; Ausgabe-Status mit <BC> und <HL> gerettet
                             ;
0142' C5                     push   bc
0143' E5                     push   hl
0144' CD 0000#               call   ?cost
0147' 18 05                  jr     ci2
0149'                cil:
                             ;
                             ; Eingabe mit <BC> und <HL> gerettet
                             ;
0149' C5                     push   bc
014A' E5                     push   hl
014B' CD 0000#               call   ?ci
014E' E1             ci2:    pop    hl
014F' C1                     pop    bc
0150' B7                     or     a                  ; Setze Flags (fuer Status)
0151' C9                     ret
                             ;
                             ; ********************************
                             ; Eingabestatus
                             ; ********************************
                             ;
                             ;
                             ; ****************
                             ; Konsole
                             ; ****************
                             ;
0152'                const:
                             ;
                             ; Konsolen Eingabe-Status. <A>=FF wenn das erste DEVICE
                             ; bereit ist, sonst <a>=00 - [ wer zuerst kommt ......]
                             ;
0152' 2A 0000#               ld     hl,(@civec)        ; Zeiger auf CON-Status BIT-Vektor
0155' 18 03                  jr     ist$scan
                             ;
                             ; ****************
                             ; AUX
                             ; ****************
                             ;
0157'                auxist:
                             ;
                             ; AUX Eingabe-Status ( sonst wie CONST )
```

```
                                ;
0157' 2A 0000#          ld      hl,(@aivec)        ; Zeiger auf AUX-Status Bit-vektor
015A'              ist$scan:
015A' 06 00             ld      b,0                ; Start mit DEVICE 0
015C'              cis$next:
015C' 29                add     hl,hl              ; MSB von Bit-Vektor in CARRY
015D' 3E 00             ld      a,0                ; Annehmen DEV ist nicht fertig
015F' DC 013B'          call    c,cistl            ; ueberpruefe Status wenn DEV vorhanden
0162' B7                or      a                  ; Flag setzen
0163' C0                ret     nz                 ; ... OK DEVICE bereit
0164' 04                inc     b                  ; naechstes DEVICE
0165' 7C                ld      a,h
0166' B5                or      l                  ; fertig ?
0167' C2 015C'          jp      nz,cis$next        ; ... wenn nicht ... sonst Schleife
016A' AF                xor     a                  ; A=0 bedeutet nicht fertig ...
016B' C9                ret

                        ;
                        ; ********************************
                        ; Zeicheneingabe
                        ; ********************************
                        ;
                        ;
                        ; ****************
                        ; Konsole
                        ; ****************
                        ;

016C'              conin:

                        ;
                        ; Konsolen Eingabe
                        ; Bringt Zeichen vom ersten fertigen DEVICE zurueck in <A>
                        ; Es wird mit jedem Aufruf dieses Programmsegmentes gleichzeitig
                        ; der Lautwerksmotor abgeschaltet
                        ;

016C' CD 0000#          call    motoff

016F' 2A 0000#          ld      hl,(@civec)
0172' 18 03             jr      in$scan

                        ;
                        ; ****************
                        ; AUX
                        ; ****************
                        ;

0174'              auxin:

                        ;
                        ; AUX Eingabe
                        ; wie conin fuer AUX
```

```
                             ;

0174' 2A 0000#               ld      hl,(@aivec)

0177'           in$scan:                                  ; Abfrage ob bereit

0177' E5                     push    hl                   ; Rette BIT-Vektor
0178' 06 00                  ld      b,0                  ; Start mit DEVICE 0

017A'           ci$next:                                  ; Schleife

017A' 29                     add     hl,hl                ; MSB in CARRY
017B' 3E 00                  ld      a,0                  ; Z-Flag fuer unbelegte Devices
017D' DC 013B'               call    c,cistl              ; hat DEVICE ein Zeichen vorliegen ?
0180' B7                     or      a                    ; Setze flags
0181' 20 08                  jr      nz,ci$ready          ; JA, Zeichen liegt vor
0183' 04                     inc     b                    ; naechste Einheit
0184' 7C                     ld      a,h                  ; ... oder bereits fertig ?
0185' B5                     or      l
0186' 20 F2                  jr      nz,ci$next           ; ... wenn nicht
0188' E1                     pop     hl                   ; mit original Bit-Vektor...
0189' 18 EC                  jr      in$scan              ; ... alles von vorne bis Zeichen ..

018B' E1        ci$ready:pop         hl
018C' C3 0000#               jp      ?ci


                             ;
                             ; ********************************
                             ; Nutzbare Unterprogramme
                             ; ********************************
                             ;

                             ;
                             ; ***************
                             ; Stringausgabe
                             ; ***************
                             ;

018F'           ?pmsg:

                             ;
                             ; Ausgabe eines Textstrings ab <HL> bis (HL)=00
                             ; oder Bit7 gesetzt - <BC> und <DE> gerettet
                             ;

018F'           pmsg$loop:                                ; Schleife ...

018F' 7E                     ld      a,(hl)               ; Zeichen einlesen
0190' E6 7F                  and     7fh                  ; maskiere Bit 7
0192' B7                     or      a                    ; 0 ?
0193' C8                     ret     z                    ; ... dann fertig
0194' CD 0131'               call    col                  ; an Ausgabeeinheit uebergeben
0197' BE                     cp      (hl)                 ; gleich?
0198' C0                     ret     nz                   ; ...wenn Bit7 gesetzt dann ungleich...
0199' 23                     inc     hl                   ; Zeiger auf naechstes Zeichen
019A' 18 F3                  jr      pmsg$loop            ; Schleife bis Text ausgegebenf

                             ;
```

```
                              ; ****************
                              ; Zahlenausgabe
                              ; ****************
                              ;

 019C'                ?pdec:

                              ;
                              ; Ausgabe einer Zahl in <HL>
                              ; im Bereich 0...65535 in binaer
                              ;

 019C' 01 01BC'               ld      bc,table10          ; Um'rechnungstabelle'
 019F' 11 D8F0                ld      de,-10000           ; Start mit zehntausender ...
 01A2' 3E 2F          next:   ld      a,'0'-1
 01A4' E5             pdecl:  push    hl                  ; Zahl retten
 01A5' 3C                     inc     a                   ; hochzaehlen Zahl in Akku
 01A6' 19                     add     hl,de               ; bis ausserhalb (DE)
 01A7' 30 04                  jr      nc,stoploop
 01A9' 33                     inc     sp                  ; Ausgleich push HL
 01AA' 33                     inc     sp
 01AB' 18 F7                  jr      pdecl

 01AD'                stoploop:

 01AD' CD 0131'               call    col                 ; Zeichen soweit ausgeben

 01B0'                next$digit:

 01B0' E1                     pop     hl                  ; Zahlenwert ruecklesen ( soweit )
 01B1' 0A                     ld      a,(bc)              ; Vergleichswert aus Tabelle
 01B2' 5F                     ld      e,a                 ; nach <DE> einlesen
 01B3' 03                     inc     bc
 01B4' 0A                     ld      a,(bc)
 01B5' 57                     ld      d,a
 01B6' 03                     inc     bc
 01B7' 7B                     ld      a,e                 ; evtl schon fertig ?
 01B8' B2                     or      d
 01B9' 20 E7                  jr      nz,next             ; ... wenn noch nicht fertig ...
 01BB' C9                     ret

 01BC'                table10:

                              ;
                              ; Um'rechnungstabelle'
                              ;

 01BC' FC18  FF9C             dw      -1000,-100,-10,-1,0

                              ;
                              ; ****************
                              ; Fehlermeldungen
                              ; ****************
                              ;

 01C6'                ?pderr:

                              ;
```

```
                                ; Fehlermeldungen (BDOS-ERROR)
                                ;

     01C6' 21 00A7"             ld      hl,drive$msg            ; Fehler 'Kopf'
     01C9' CD 018F'             call    ?pmsg                   ;  senden
     01CC' 3A 01FF'             ld      a,(@drv)                ; aktuelle Lautwerksnummer
     01CF' C6 41                add     a,'A'                   ; in A...P wandeln
     01D1' CD 0131'             call    col                     ; senden
     01D4' 21 00B4"             ld      hl,track$msg            ; danach TRACK-Nummer
     01D7' CD 018F'             call    ?pmsg
     01DA' 2A 0200'             ld      hl,(@trk)
     01DD' CD 019C'             call    ?pdec                   ; in DECIMAL
     01E0' 21 00E8"             ld      hl,sector$msg           ; desgl. mit SEKTOR-Nummer
     01E3' CD 018F'             call    ?pmsg
     01E6' 2A 0202'             ld      hl,(@sect)
     01E9' CD 019C'             call    ?pdec
     01EC' C9                   ret

                                ;
                                ; ****************
                                ; Bankselect
                                ; ****************
                                ;

     01ED'              bnksel:

                                ;
                                ; Bankselect - Auswahl der CPU-Bank fuer weitern Programmablauf
                                ;

     01ED' 32 0208'             ld      (@cbnk),a               ; als momentan Bank speichern
     01F0' C3 0000#             jp      ?bank                   ; dann Bank anwaehlen

                                ;
                                ; ****************
                                ; XON-XOFF Tabelle
                                ; ****************
                                ;

     01F3'              xofflist:

                                ;
                                ; RAM-Simulation des Ein/Aus-Status bei XON/XOFF Protokoll
                                ; CNTRL-S macht 0 aus FF
                                ;

                                rept    12                      ; nur 12 Devicees moeglich !

                                db      -1

                                endm
     01F3' FF           A       db      -1
     01F4' FF           A       db      -1
     01F5' FF           A       db      -1
     01F6' FF           A       db      -1
     01F7' FF           A       db      -1
     01F8' FF           A       db      -1
     01F9' FF           A       db      -1
```

```
01FA' FF        A      db     -1
01FB' FF        A      db     -1
01FC' FF        A      db     -1
01FD' FF        A      db     -1
01FE' FF        A      db     -1

                       dseg                          ; weiter in Bank 0

                       ;
                       ; ********************************
                       ; DISK I/O Interface Routinen
                       ; ********************************
                       ;
                       ;
                       ; in diesen Programmteilen wird jedoch NICHT physikalisch
                       ; auf die Laufwerke zugegriffen ...
                       ;

                       ;
                       ; ***************
                       ; Diskselekt
                       ; ***************
                       ;

0034'           seldsk:

                       ;
                       ; Anwahl eines Laufwerkes. Laufwerksnummer in <C>
                       ; Veranlasst LOGIN wenn erstmals angesprochen
                       ; Bringt in <HL> Adresse des DPH's zurueck (Disk Parameter Header)
                       ; Ist BIT 0 in DE=0 verlangt CP/M ein erneutes Login
                       ; ( z.B. nach Diskettenwechsel )
                       ;

0034" 79               ld     a,c            ; Laufwerks-Nummer nach <A>
0035" 32 01FF'         ld     (@drv),a       ; und in RAM ablegen
0038" 69               ld     l,c            ; danach Nummer in <HL>
0039" 26 00            ld     h,0            ; kopieren
003B" 29               add    hl,hl          ; *2 Index in XDPH-Tabelle
003C" 01 0000#         ld     bc,@dtbl       ; Zeiger in Laufwerkstabelle
003F" 09               add    hl,bc          ; errechne Adr des XDPH fuer Laufwerk
0040" 7E               ld     a,(hl)
0041" 23               inc    hl
0042" 66               ld     h,(hl)         ; XDPH-Adresse nach HL
0043" 6F               ld     l,a
0044" B4               or     h              ; Laufwerk 'vorhanden'
0045" C8               ret    z              ; .... nein, nicht implementiert
0046" CB 43            bit    0,e            ; LOGIN verlangt ?
0048" C0               ret    nz             ; ... wenn bereits selektiert
0049" E5               push   hl             ; Rette Adresse des XDPH
004A" EB               ex     de,hl          ; ... auf Stack und in <DE>
004B" 21 FFFA          ld     hl,-6          ; Berechne Zeiger auf LOGIN
004E" CD 0098"         call   getadr
0051" CD 008A"         call   ipchl          ; LOGIN ausfuehren
0054" E1               pop    hl             ; XDPH-Zeiger
0055" C9               ret

                       ;
```

```
                                ; ****************
                                ; HOME
                                ; ****************
                                ;

0056"                   .home:

                                ;
                                ; veranlasst das Laufwerk (nach Ansprache) auf TRACK 0 vorzu-
                                ; gefahren, damit die Stellung des Schreib/Lesekopfes
                                ; bakannt ist
                                ;

0056" 01 0000                   ld      bc,0

                                ;
                                ; ****************
                                ; Track setzen
                                ; ****************
                                ;

0059"                   settrk:

                                ;
                                ; Die Variable (@TRK) wird mit der gewuenschten Track-Nummer
                                ; - in <BC> - ueberschrieben. Diese Track-Nummer wird beim
                                ; naechsten physikalischen Zugriff auf ein Laufwerk angewaehlt
                                ;

0059" ED 43 0200'               ld      (@trk),bc
005D" C9                        ret

                                ;
                                ; ****************
                                ; Sektor setzen
                                ; ****************
                                ;

005E"                   setsec:

                                ;
                                ; wie SETTRK jedoch fuer den naechsten Sektor
                                ;

005E" ED 43 0202'               ld      (@sect),bc
0062" C9                        ret

                                ;
                                ; ****************
                                ; RAM-Adresse setzen
                                ; ****************
                                ;

0063"                   setdma:

                                ;
                                ; die RAM-Adresse wo der naechste Sektor hingelesen wird, wird
                                ; in der Variablen (@DMA) abgelegt, Uebergabe dieser Adresse
```

```
                              ; in <BC>. Gleichzeitig wird @UBNK (Bank wohin geschrieben wird)
                              ; als @CBNK (Bank die gerade angewaehlt ist) selektiert.
                              ;
0063" ED 43 0204'             ld      (@dma),bc
0067" 3A 0208'                ld      a,(@cbnk)
                              ;
                              ; ****************
                              ; Bank waehlen
                              ; ****************
                              ;
006A"                 setbnk:
                              ;
                              ; setzt Variable @UBNK( Bank wohin geschrieben wird ) auf den
                              ; in <A> uebergebenen Wert
                              ;
006A" 32 0207'                ld      (@ubnk),a
006D" C9                      ret
                              ;
                              ; ****************
                              ; Sektor SKEW
                              ; ****************
                              ;
006E"                 sectrn:
                              ;
                              ; Berechnet  aus der logischen Sektor-Nummer die physikalische
                              ; Sektor-Nummer ( falls SKEW vorgegeben !). Logischer Sektor
                              ; wird in <BC> uebergeben, physikalischer Sektor in <HL> zurueck-
                              ; erwartet. Ist SKEW=0 wird der log-Sektor als phys-Sektor ueber-
                              ; geben. Der Index in die SKEW-Tabelle (XLTxx) wird in DE ueber-
                              ; geben. Info ueber SKEW steht in jedem XDPH.
                              ;
006E" 69                      ld      l,c                 ; Kopie nach HL
006F" 60                      ld      h,b
0070" 7A                      ld      a,d                 ; SKEW=0 ?
0071" B3                      or      e
0072" C8                      ret     z                   ; dann fertig
0073" EB                      ex      de,hl               ; sonst Offset in SKEW-Tabelle
0074" 09                      add     hl,bc               ; berechnen
0075" 6E                      ld      l,(hl)              ; und Wert nach HL
0076" 26 00                   ld      h,0                 ; SKEW nicht > 255 !
0078" C9                      ret
                              ;
                              ; ****************
                              ; Sektor lesen
                              ; ****************
                              ;
0079"                 read:
```

```

                                ;
                                ; holt aus dem XDPH die Adresse der gueltigen LESE-Routine
                                ; und springt diese an
                                ;

0079" CD 008B"                  call    getdph                  ; Hole Adresse des XDPH
007C" 21 FFF8                   ld      hl,-8                   ; Offset zum Leservektor
007F" 18 06                     jr      rw$common               ; dann ausfuehren

                                ;
                                ; ****************
                                ; Sektor schreiben
                                ; ****************
                                ;

0081"                   write:

                                ;
                                ; holt aus dem XDPH die Adresse der gueltigen SCHREIB-Routine
                                ; und springt diese an
                                ;

0081" CD 008B"                  call    getdph                  ; Hole Adresse des XDPH
0084" 21 FFF6                   ld      hl,-10                  ; Offset zum  Schreibvektor

0087"                   rw$common:

                                ;
                                ; gemeinsamer Programmteil READ/WRITE ausfuehren
                                ;

0087" CD 0096"                  call    getadr                  ; Adresse der Funktion
008A" E9                ipchl:  jp      (hl)                    ; Ausfuehren READ oder WRITE

                                ;
                                ; ****************
                                ; Hilfsprogramme
                                ; ****************
                                ;

008B"                   getdph:

                                ;
                                ; berechnet nach der Laufwerksnummer in der Variablen (@DRV)
                                ; die Adresse des entsprechenden XDPH und uebergibt diese
                                ; Adresse in <DE>
                                ;

008B" E5                        push    hl                      ; Benutzerregister retten
008C" 2A 01FF'                  ld      hl,(@drv)               ; Laufwerksnummer
008F" 26 00                     ld      h,0
0091" 29                        add     hl,hl                   ; *2
0092" 11 0000#                  ld      de,@dtbl                ; Zeiger auf Laufwerkstabelle
0095" 19                        add     hl,de
0096" 5E                        ld      e,(hl)
0097" 23                        inc     hl
0098" 56                        ld      d,(hl)
```

```
0099" E1                pop     hl                      ; zurueck Benutzerregister
009A" C9                ret
009B"           getadr:
009B" 19                add     hl,de
009C" 7E                ld      a,(hl)
009D" 23                inc     hl
009E" 66                ld      h,(hl)
009F" 6F                ld      l,a
00A0" C9                ret
                        ;
                        ; ****************
                        ; (De)-Blocking
                        ; ****************
                        ;
00A1"           multio:
                        ;
                        ; setzt Sektor-Zaehler
                        ; wenn die Sektoren groesser als 128 byte (Standard
                        ; CP/M) sind, wie dies bei doppelter Schreibdichte (dd) ueblich
                        ; ist. Das Blocking und Deblocking wird entsprechend der
                        ; Sektorgroesse von CP/M uebernommen.
                        ; Uebergabe in <A>
                        ;
00A1" 32 0206'          ld      (@cnt),a
00A4" C9                ret
                        ;
                        ; ****************
                        ; Flush
                        ; ****************
                        ;
00A5"           flush:
                        ;
                        ; BIOS blocking buffer flush
                        ; NICHT IMPLEMENTIERT CP/M uebernimmt blocking/deblocking
                        ;
00A5" AF                xor     a                       ; 0-Fehler
00A6" C9                ret
                        ;
                        ; ****************
                        ; Fehlertexte
                        ; ****************
                        ;
00A7"           drive$msg:
00A7" 0D 0A 46 65       db      cr,lf,'Fehler auf',' '+80h
```

```
1161 00B4"            track$msg:
1162
1163 00B4" 3A 20 54 AD         db      ': T','-'+80h
1164
1165 00B8"            sector$msg:
1166
1167 00B8" 2C 20 53 AD         db      ', S','-'+80h
1168
1169                          ;
1170                          ; ***************
1171                          ; Laufwerks-Variable
1172                          ; ***************
1173                          ;
1174
1175                          cseg
1176
1177 01FF'   0001   @drv:     defs    1          ; Angewaehltes Laufwerk
1178 0200'   0002   @trk:     defs    2          ; gueltige Track Nummer
1179 0202'   0002   @sect:    defs    2          ; gueltige Sektor Nummer
1180 0204'   0002   @dma:     defs    2          ; gueltige DMA Adresse
1181 0206' 00       @cnt:     defb    0          ; record-Zaehler bei multisector Transf
1182
1183                          cseg
1184
1185 0207' 00       @dbnk:    defb    0          ; Bank fuer DMA Operationen
1186 0208' 00       @cbnk:    defb    0          ; Bank fuer Programmablauf ( in TPA )
1187
1188                          ;
1189                          ; ENDE des invariablen Teiles des BIOS
1190                          ;
1191
1192                          end
0 Error(s) Detected. 521 Program Bytes. 188 Data Bytes.
250 Symbols Detected.
```



```
01A4' PDECL          770   776
018F' PMSG$LOOP      742   752
0079" READ           118  1044
0087" RW$COMMON     1053  1071
00B8" SECTOR$MSG     825  1165
006E" SECTRN         122  1016
0034" SELDSK         113   895
0078' SET$JUMPS      259   277   292
006A" SETBNK         136  1000
0063" SETDMA         117   982
005E" SETSEC         116   967
0059" SETTRK         115   950
01AD' STOPLOOP       773   778
01BC' TABLE10        767   796
00B4" TRACK$MSG      821  1161
006C' WBOOT          104   268
0081" WRITE          119  1061
01F3' XOFFLIST       540   852
```

```
0015' ?AUXI        81   111
0036' ?AUXIS       84   124
0012' ?AUXO        81   110
0039' ?AUXOS       84   125
01F1# ?BANK        61   844
0051' ?BNKSL       85   135   302
0000' ?BOOT        81   103
018D# ?CI          42   601   721
000A# ?CINIT       43   128   201
013E# ?CIST        42   579
00D7# ?CO          42   420
0009' ?CONIN       81   106
000C' ?CONO        81   107   567
0033' ?CONOS       84   123
0006' ?CONST       81   105
0145# ?COST        42   534   590
003F' ?DEVIN       84   128
0042' ?DRTBL       84   129
003C' ?DVTBL       84   127
0048' ?FLUSH       85   131
0018' ?HOME        82   112
0004# ?INIT        35   188
0067# ?LDCCP       36   265
000F' ?LIST        81   109
002D' ?LISTS       83   121
0045' ?MLTIO       85   130
004B' ?MOV         85   133
004C# ?MOVE        60   133
019C' ?PDEC        73   760   824   828
01C6' ?PDERR       74   810
018F' ?PMSG        73   735   817   822   826
0027' ?READ        82   118
0073# ?RLCCP       36   283
0030' ?SCTRN       83   122
001B' ?SLDSK       82   113
0054' ?STBNK       85   136
0024' ?STDMA       82   117
0021' ?STSEC       82   116
001E' ?STTRK       82   115
004E' ?TIM         85   134
004F# ?TIME        67   134
0003' ?WBOOT       81   104   309
002A' ?WRITE       82   119
0057' ?XMOV        85   137
0058# ?XMOVE       60   137
0056" .HOME       112   934
0175# @AIVEC       27   641   699
00E8# @AOVEC       27   374   467
0000# @BNKBF       29
0208' @CBNK        59   843   992  1186
0170# @CIVEC       27   626   683
0206' @CNT         53  1132  1181
00E3# @COVEC       27   358   452
010B# @CTBL        44   331   529
0207' @DBNK        53  1007  1185
0204' @DMA         53   991  1180
01FF' @DRV         52   818   906  1095  1177
0093# @DTBL        51   217   334   910  1098
```

Cross Reference:

```
00ED# @LOVEC              27    390    482
008C# @MXTPA              28    311
0202' @SECT               52    827    973   1179
0200' @TRK                52    823    958   1178
0174' AUXIN              111    692
0157' AUXIST             124    635
00E7' AUXOST             125    461
00BF' AUXOUT             110    367
FFFF  BANKED             299
01ED' BNKSEL             135    837
0000" BOOT               103    168
0063' BOOT$1             247    251
00B2' BOOT$STACK         175    276    323
0008" C$INIT$LOOP        198    204
0100  CCP                266    284
017A' CI$NEXT            706    716
018B' CI$READY           712    720
0149' CI1                544    593
014E' CI2                569    580    591    602
015C' CIS$NEXT           647    657
013B' CIST1              542    571    651    710
00C9' CO$NEXT            406    429
0131' CO1                557    748    780    820
016C' CONIN              106    672
00E2' CONOST             123    445
00BA' CONOUT             107    351
0152' CONST              105    619
00F1' COS$NEXT           493    506
0142' COST1              553    582
0103' COSTER             415    498    516
000D  CR                1159
0011  CTLQ               545
0013  CTLS               549
0016" D$INIT$LOOP        219    246
002E" D$INIT$NEXT        228    243
00B2' DEVTBL             127    331
00A7" DRIVE$MSG          816   1157
00A5" FLUSH              131   1141
009B" GETADR             923   1077   1106
008B" GETDPH            1051   1068   1096
00B6' GETDRV             129    334
0177' IN$SCAN            684    701    718
008A" IPCHL               75    240    924   1078
015A' IST$SCAN           627    643
000A  LF                1159
00C4' LIST               109    383    383
00EC' LISTST             121    476
0010  MB$XON$XOFF        532
016D# MOTOFF              50    681
00A1" MULTIO             130   1121
01A2' NEXT               769    793
01B0' NEXT$DIGIT         782
00DB' NOT$OUT$DEVICE     409    424
00CE' NOT$OUT$READY      413    417
0126' NOT$Q              546    549
012B' NOT$S              550    552
00EF' OST$SCAN           453    468    484
00C7' OUT$SCAN           359    375    392
```

Kapitel 7 Das BIOS-Segment CHARIO

Dieses Programm-Segment des BIOS ist für die reine Hardware-Anpassung der Zeichen- (Character) Ein- und Ausgaben verantwortlich. Es sind hierunter die Treiber zu verstehen, die Zeichen bearbeiten, die nicht auf Diskette gespeichert sind oder werden, also alles was mit der Konsole, dem Drucker oder dem AUX-Kanal zu tun hat.

Das Segment CHARIO besteht prinzipiell aus 7 Teilen:

1. Die DEVICE (phys. Einheit)-Tabelle. In dieser Tabelle sind alle vorgesehenen physikalischen Einheiten eingetragen. Ihre Reihenfolge untereinander ist das Kriterium wie die einzelnen Treiber angesprochen werden.

   Im Beispiel wird mit DEVICE 0 immer die physikalische Einheit 'KEY' (Tastatur) angesprochen und mit DEVICE 2 die physikalische Einheit 'TERM' (wie Terminal).

2. Abhängig von der DEVICE-Tabelle: die Initialisierungsunterprogramme aller Einheiten.

3. Abhängig von der DEVICE-Tabelle: alle Ausgabe-Status-Unterprogramme.

4. Abhängig von der DEVICE-Tabelle: alle Eingabe-Status-Unterprogramme.

5. Abhängig von der DEVICE-Tabelle: alle Ausgabetreiber.

6. Abhängig von der DEVICE-Tabelle: alle Eingabetreiber.

7. Entsprechende Verteiler-Unterprogramme auf die einzelnen Treiberprogramme.

Die Zuweisung der entsprechenden physikalischen Einheit(en) an die fünf möglichen logischen Einheiten erfolgt im Programmsegment BOOT oder mit dem TPA-Befehl DEVICE.COM.

Die 5 möglichen logischen Einheiten sind:

CONIN, CONOUT	für die Konsole
AUXIN, AUXOUT	für die Zusatzschnittstelle
LST	für den Drucker

Jede beliebige physikalische Eingabe-Einheit kann jeder beliebigen logischen Eingabe-Einheit zugewiesen werden, das gleiche gilt auch für die Ausgabe-Einheiten.

Es können einer logischen Einheit mehrere (theoretisch bis zu 16) physikalische Einheiten zugeordnet werden.

Die Zuweisung erfolgt durch setzen der entsprechenden Zuweisungsbits (gesetzt=1, nicht gesetzt=0) in den entsprechenden 16 Bit-Vektoren CIVEC..LOVEC im SCB. (Näheres dazu siehe auch Kapitel 10 - BOOT).

Sind mehrere Ausgabeeinheiten vorgesehen, so wird auf jede Einheit ausgegeben, das Tempo bestimmt dann natürlich die langsamste Einheit. Im Falle der Eingabe wird immer nur die Eingabe berücksichtigt, die zuerst erfolgt.

Das nachfolgende Listing zeigt das Programmsegment in allen Details.

Dieses Listing wird im allgemeinen das 'Hauptarbeitsfeld' des CP/M Benutzers sein, der in 'seinem' System neue und neuartige Ein- Ausgabe-Schnittstellen implementieren will.

In dieses Programmsegment hinein gehören auch die speziellen Treiber für spezielle Video-Karten wie z.B. der Treiber für die VIDEO-80 Karte von ELZET-80 Mikrocomputer.

Derartige Treiber sind meist recht komplex, da sie die Verwaltung eines Zeichenausgabepuffers in allen Details übernehmen müssen.

Um 'Platz' in der TPA zu schaffen, können die Treiber, nach entsprechender Bank- und Stackumschaltung natürlich auch in den Bankbereich eines CP/M 3 Systems verlegt werden. Bei 'Banküber-greifenden' Softwaretreiber muß u.U. ein 'eigener' Stack im gemeinsamen (Common) Speicherbereich vorgesehen werden, da der (evtl. ursprünglich in der TPA liegende) Stackpointer zwar immer noch auf die richtige Adresse, aber in der falschen Bank zeigt. Dies kann zu 'ominösen' Systemabstürzen führen, die recht schlecht zu finden sind .....

```
                                MACLIB  DEFAULT.INC     ; Vereinbarungen
                                MACLIB  MODEBAUD.INC
                        ;*****************************************************
                        ;*                                                   *
                        ;* rel 1.1 vom 31.10.85                              *
                        ;*                                                   *
                        ;* physikalischer Treiber fuer Zeichen Ein/Ausgabe   *
                        ;*                                                   *
                        ;* geschrieben von RAOUL O. KOERBER                  *
                        ;* nach Unterlagen von DRI teilweise mit grossen     *
                        ;* Abaenderungen um das Programm flexibler an die    *
                        ;* Moeglichkeiten eines NDR-Computers anzupassen     *
                        ;*                                                   *
                        ;*****************************************************

                                ;
                                ; ***************
                                ; Systemadressen
                                ; ***************
                                ;

                                public  ?cinit,?ci,?co,?cist,?cost
                                public  @ctbl
                                public  tbaud
                                extrn   inisub

                                cseg                            ; im gemeinsamen RAM-Bereich

                                ;
                                ; ******************************
                                ; Initialisierung
                                ; ******************************
                                ;

0000'                   ?cinit:

                                ;
                                ; Diese Routine wurde anders als der DRI Vorschlag
                                ; aufgebaut um eine groessere Flexibilitaet zu erreichen
                                ; erwartet beim Aufruf eine Device-Nummer in <C>
                                ; uebergibt dem angesprungenen Unterprogramm
                                ; <C> unveraendert und <DE> mit Zeiger auf @CTBL+7 (baudrate)
                                ;

0000' 79                        ld      a,c                     ; Device-Nummer
0001' FE 06                     cp      max$devices             ; noch im Bereich des Moeglichen ?
0003' D0                        ret     nc                      ; ... wenn nicht
0004' 69                        ld      l,c                     ; Device-Nummer als
0005' 26 00                     ld      h,0                     ; Zaehler
0007' 29                        add     hl,hl                   ; *2
0008' 11 0119'                  ld      de,my$det               ; Zeiger in DEVICE-Tabelle
000B' 19                        add     hl,de
000C' 7E                        ld      a,(hl)                  ; Adresse auslesen
000D' 23                        inc     hl
000E' 66                        ld      h,(hl)
000F' 6F                        ld      l,a
0010' 11 012C'                  ld      de,@ctbl+7              ; Zeiger in @CTBL
0013' E9                        jp      (hl)                    ; und Routine ausfuehren
```

```

                                ;
                                ; ***********************************************
                                ; Hier folgen die einzelnen Initroutinen
                                ; soll keine Neuinitialisierung erfolgen
                                ; kann die Routine einfach mit RET abgeschlossen werden
                                ; ***********************************************
                                ;

                                ;
                                ; **** SER als Terminalanschluss
                                ;

 0014' 3A 013C'      i$dev2:  ld     a,(tbaud)         ; Baudrate einlesen
 0017' 47                     ld     b,a               ; nach (B)
 0018' 21 0027'               ld     hl,i$d2msg+2      ; Zeiger auf baudrate-Init
 001B' 7E            i$dev21: ld     a,(hl)            ; Wert einlesen und
 001C' E6 F0                  and    11110000b         ; 'alte' Baudrate maskieren
 001E' B0                     or     b                 ; und 'neue' Baudrate dazufuegen
 001F' 77                     ld     (hl),a            ; wieder ablegen
 0020' 2B                     dec    hl
 0021' 2B                     dec    hl                ; Zeiger auf Laenge des INIT-Strings
 0022' C3 0000#               jp     inisub

 0025' 02            i$d2msg: db     2                 ; Laenge
 0026' FF 9E                  db     ucon,10011110b    ; 2stop - 8bit + Baudrate
 0028' FE 0B                  db     ucmd,00001011b    ; keine Paritaet -dtr

                                ;
                                ; **** AUX mit SER
                                ;

 002A' 3A 0144'      i$dev3:  ld     a,(abaud)
 002D' 47                     ld     b,a
 002E' 21 0035'               ld     hl,i$d3msg+2
 0031' 18 E8                  jr     i$dev21

 0033' 02            i$d3msg: db     2
 0034' CF 9E                  db     ucon2,10011110b
 0036' CE 0B                  db     ucmd2,00001011b

                                ;
                                ; **** SPRINT mit SER
                                ;

 0038' 3A 014C'      i$dev4:  ld     a,(pbaud)
 003B' 47                     ld     b,a
 003C' 21 0043'               ld     hl,i$d4msg+2
 003F' 18 DA                  jr     i$dev21

 0041' 02            i$d4msg: db     2
 0042' EF 9E                  db     ucon1,10011110b
 0044' EE 0B                  db     ucmd1,00001011b

                                ;
                                ; **** PPRINT mit CENT
                                ;

 0046' C9            i$dev5:  ret                      ; kein INIT noetig
```

```
                           ;
                           ; ********************************
                           ; Beginn der Ein/Ausgaberoutinen
                           ; ********************************
                           ;
                           ; Diese wurden entgegen dem DRI-Vorschlag ebenfalls
                           ; in einzelne Unterprogramme aufgesplittet um auch
                           ; hier maximale Flexibilitaet zu erreichen - wenn
                           ; dies auch mit weniger optimalem CODE erreicht wird
                           ;
                           ; Einsprung in die Unterprogramme erfolgt immer
                           ; ueber eine entsprechende Tabelle mit dem Device-Offset
                           ; ( von CP/M  uebergeben ) in <B>

                           ;
                           ; ***************
                           ; Eingabe
                           ; ***************
                           ;

0047'               ?ci:

                           ;
                           ; Konsolen Eingabe ( alle Devices )
                           ; Device-Nummer in <B>
                           ; Zurueck mit Zeichen in <A>
                           ; Falls Device nicht angesprochen werden kann, ist A=1AH
                           ;

0047' 78                   ld      a,b                 ; Device-Nummer
0048' FE 06                cp      max$devices
004A' 3E 1A                ld      a,1ah
004C' D0                   ret     nc                  ; physikalisch nicht vorhanden
004D' CD 005D'      ?ci1:  call    ?cist1              ; warte bis Device bereit
0050' 28 FB                jr      z,?ci1              ;

                           ;
                           ; ?CIST bringt Device-Offset in <DE> zurueck
                           ;

0052' 21 00E9'             ld      hl,ci$tbl
0055' 18 28                jr      ?cost2              ; Rest von dort ...

                           ;
                           ; ***************
                           ; Ein-Status
                           ; ***************
                           ;

0057'               ?cist:

                           ;
                           ; Konsolen Eingabe-Status
                           ; Device-Nummer in <B>
                           ; Zurueck mit <A>=FF wenn Device bereit
                           ; sonst <A>=00 und Z-Flag gesetzt
                           ; Zurueck mit <DE> bereits als Offset
                           ;
```

```

0057' 78                    ld      a,b                     ; Device-Nummer
0058' FE 06                 cp      max$devices
005A' 3E 00                 ld      a,0                     ; nicht ansprechbar ?
005C' D0                    ret     nc                      ; wenn physikalisch nicht vorhanden
005D' 58           ?cist1:  ld      e,b                     ; Displacement berechnen
005E' 16 00                 ld      d,0
0060' 21 00F5'              ld      hl,cist$tbl
0063' 18 1A                 jr      ?cost2                  ; dort geht's weiter


                            ;
                            ; ****************
                            ; Ausgabe
                            ; ****************
                            ;

0065'              ?co:


                            ;
                            ; Konsolen Ausgabe
                            ; Device-Nummer in <B> und auszugebendes Zeichen in <C>
                            ;

0065' 78                    ld      a,b                     ; Device-Nummer
0066' FE 06                 cp      max$devices
0068' D0                    ret     nc                      ; physikalisch nicht vorhanden
0069' CD 0079'     ?co1:    call    ?cost1                  ; Status lesen
006C' 28 FB                 jr      z,?co1                  ; warten bis bereit

                            ;
                            ; ?COST bringt Offset in <DE> mit
                            ;

006E' 21 0101'              ld      hl,co$tbl
0071' 18 0C                 jr      ?cost2                  ; ... Rest dort

                            ;
                            ; ****************
                            ; Aus-Status
                            ; ****************
                            ;

0073'              ?cost:


                            ;
                            ; Konsolen Ausgabe-Status
                            ; Device-Nummer in <B>
                            ; Zurueck mit <A>=FF wenn bereit
                            ; sonst <A>=00
                            ; Zurueck mit <DE> bereits als Offset
                            ; ACHTUNG <C> darf nicht veraendert werden !!!!
                            ;

0073' 78                    ld      a,b                     ; Device-Nummer
0074' FE 06                 cp      max$devices
0076' 3E 00                 ld      a,0                     ; ... Fehler
0078' D0                    ret     nc                      ; wenn physikalisch nicht vorhanden
0079' 58           ?cost1:  ld      e,b
007A' 16 00                 ld      d,0
```

```
007C' 21 0100'                ld      hl,cost$tbl
007F' 19            ?cost2:   add     hl,de
0080' 19                      add     hl,de                 ; 2*
0081' 7E                      ld      a,(hl)
0082' 23                      inc     hl
0083' 66                      ld      h,(hl)
0084' 6F                      ld      l,a
0085' E9                      jp      (hl)

                              ;
                              ; ***************
                              ; Eingabe Unterprogramme
                              ; ***************
                              ;

0086'               _ci2:

                              ;
                              ; Terminal (SER) Eingabe
                              ;

0086' CD 009A'                call    _cist2
0089' 28 FB                   jr      z,_ci2
008B' DB FC                   in      a,(udat)
008D' E6 7F                   and     7fh
008F' C9            dummy:    ret

0090'               _ci3:

                              ;
                              ; AUX (SER) Eingabe
                              ;

0090' CD 00A2'                call    _cist3
0093' 28 FB                   jr      z,_ci3
0095' DB CC                   in      a,(udat2)
0097' E6 7F                   and     7fh
0099' C9                      ret

                              ;
                              ; ***************
                              ; Ein-Status UP
                              ; ***************
                              ;

009A'               _cist2:

                              ;
                              ; Eingabe Status SER (TERMINAL)
                              ;

009A' DB FD                   in      a,(usta)
009C' E6 08         _cist21:  and     00001000b
009E' C8                      ret     z             ; nicht fertig = 0
009F' F6 FF                   or      11111111b
00A1' C9                      ret
```

```
00A2'                  _cist3:
                                  ;
                                  ; Eingabe Status SER (AUX)
                                  ;
00A2' DB CD                       in      a,(usta2)
00A4' 18 F6                       jr      _cist21
                                  ;
                                  ; ****************
                                  ; Ausgabe UP
                                  ; ****************
                                  ;
00A6'                  _co2:
                                  ;
                                  ; Ausgabe ueber SER (TERMINAL)
                                  ;
00A6' CD 00D3'                    call    _cost2
00A9' 28 FB                       jr      z,_co2
00AB' 79                          ld      a,c
00AC' D3 FC                       out     (udat),a
00AE' C9                          ret
00AF'                  _co3:
                                  ;
                                  ; Ausgabe ueber SER (AUX)
                                  ;
00AF' CD 00DB'                    call    _cost3
00B2' 28 FB                       jr      z,_co3
00B4' 79                          ld      a,c
00B5' D3 CC                       out     (udat2),a
00B7' C9                          ret
00B8'                  _co4:
                                  ;
                                  ; Ausgabe ueber SER (SPRINT)
                                  ; vorlaeufig kein Protokoll eingebaut
                                  ;
00B8' CD 00DF'                    call    _cost4
00BB' 28 FB                       jr      z,_co4
00BD' 79                          ld      a,c
00BE' D3 EC                       out     (udat1),a
00C0' C9                          ret
00C1'                  _co5:
                                  ;
                                  ; Ausgabe ueber IO (PPRINT)
                                  ;
```

```
00C1' CD 00E3'              call    _cost5          ; bereit?
00C4' 28 FB                 jr      z,_co5          ; sonst warten
00C6' 79                    ld      a,c
00C7' D3 48                 out     (iodat),a
00C9' AF                    xor     a               ; Strobe down
00CA' D3 49                 out     (iocmd),a
00CC' 3C                    inc     a               ; Strobe up
00CD' D3 49                 out     (iocmd),a
00CF' C9                    ret

                            ;
                            ; ****************
                            ; Aus-Status UP
                            ; ****************
                            ;

00D0'               _cost1:

                            ;
                            ; Ausgabe Status GDP
                            ;

00D0' AF                    xor     a               ; ist immer fertig wenn zurueck
00D1' 3D                    dec     a
00D2' C9                    ret

00D3'               _cost2:

                            ;
                            ; Ausgabe Status SER (TERM)
                            ;

00D3' DB FD                 in      a,(usta)
00D5' E6 10         _cost21:and     00010000b
00D7' C8                    ret     z
00D8' F6 FF                 or      11111111b
00DA' C9                    ret

00DB'               _cost3:

                            ;
                            ; Ausgabe Status SER (AUX)
                            ;

00DB' DB CD                 in      a,(usta2)
00DD' 18 F6                 jr      _cost21

00DF'               _cost4:

                            ;
                            ; Ausgabe Status SER (SPRINT)
                            ;

00DF' DB ED                 in      a,(usta1)
00E1' 18 F2                 jr      _cost21
```

```
00E3'               _cost5:


                               ;
                               ; Ausgabe Status PPRINT
                               ;

00E3' DB 49                    in      a,(iocmd)       ; Status
00E5' 0F                       rrca
00E6' 3F                       ccf
00E7' 9F                       sbc     a,a             ; FF wenn bereit 0 falls nicht
00E8' C9                       ret

                               ;
                               ; ****************
                               ; Sprungtabellen
                               ; ****************
                               ;

00E9'               ci$tbl:

                               ;
                               ; Sprungtabelle ?CI
                               ;

00E9' FC03                     dw      econin          ; KEY
00EB' 008F'                    dw      dummy           ; GDP hat keine Eingabe
00ED' 0086'                    dw      _ci2            ; Terminal
00EF' 0090'                    dw      _ci3            ; AUX
00F1' 008F'                    dw      dummy           ; SSPRINT hat keine Eingabe
00F3' 008F'                    dw      dummy           ; PPRINT hat keine Eingabe

00F5'               cist$tbl:

                               ;
                               ; Sprungtabelle ?CIST
                               ;

00F5' FC00                     dw      econist         ; KEY-Status
00F7' 008F'                    dw      dummy           ; GDP hat keine Eingabe
00F9' 009A'                    dw      _cist2          ; Terminal
00FB' 00A2'                    dw      _cist3          ; AUX
00FD' 008F'                    dw      dummy           ; SPRINT hat keine Eingabe
00FF' 008F'                    dw      dummy           ; PPRINT hat keine Eingabe

0101'               co$tbl:

                               ;
                               ; Sprungtabelle ?CO
                               ;

0101' 008F'                    dw      dummy           ; KEY hat keine Ausgabe
0103' FC06                     dw      econout         ; GDP
0105' 00A6'                    dw      _co2            ; Terminal
0107' 00AF'                    dw      _co3            ; AUX
0109' 00B8'                    dw      _co4            ; SPRINT
010B' 00C1'                    dw      _co5            ; PPRINT

```

```
0100'                 cost$tbl:

                                ;
                                ; Sprungtabelle ?COST
                                ;

0100' 008F'                     dw      dummy           ; KEY hat keine Ausgabe
010F' 00D0'                     dw      _cost1          ; GDP
0111' 00D3'                     dw      _cost2          ; Terminal
0113' 00DB'                     dw      _cost3          ; AUX
0115' 00DF'                     dw      _cost4          ; SPRINT
0117' 00E3'                     dw      _cost5          ; PPRINT

0119'                 my$def:

                                ;
                                ; Zuweisung der Devices bei INIT
                                ; im Gegensatz zum DRI-Vorschlag werden bei ?CINIT
                                ; einzelne Init-Routinen aufgerufen - damit ist
                                ; wesentlich mehr Flexibilitaet verbunden
                                ; die Vektorn _devX muessen zu @CTBL in unveraender-
                                ; barem Verhaeltnis stehen um einfach darauf zu-
                                ; greifen zu koennen
                                ;

0119' 008F'           _dev0:    dw      dummy                   ; KEY - kein INIT
011B' 008F'           _dev1:    dw      dummy                   ; GDP - kein INIT
011D' 0014'           _dev2:    dw      i$dev2                  ; Terminal via SER
011F' 002A'           _dev3:    dw      i$dev3                  ; SER-AUX
0121' 0038'           _dev4:    dw      i$dev4                  ; SPRINT
0123' 008F'           _dev5:    dw      dummy                   ; PPRINT

                                ;
                                ; ****************
                                ; Device-Tabelle
                                ; ****************
                                ;
                                ; hier folgt die Devicetabelle auf die CP/M3 zugreift
                                ; sie darf daher nicht in ihrem prinzipiellen Aufbau
                                ; erweitert werden, kann jedoch bis zu 14 Einheiten
                                ; umfassen
                                ;

0125'                 @ctbl:

                                ;
                                ; Zeichen Ein/Ausgabe-Einheiten (Devices)
                                ; Reihenfolge muss mit BAUD$PORTS und DATA$PORTS
                                ; korrespondieren
                                ; Textlaenge ist vorgeschrieben -Textinhalt nicht
                                ; das nach dem Text folgende bit gibt Information
                                ; welcher Art das angesprochene Device ist, danach
                                ; folgt ein Offset in eine Baudrate-tabelle

0125' 4B 45 59 20               db      'KEY   '                ; Device 0 Tastatur
012B' 01                        db      mb$input                ; nur Eingabe
012C' 00                        db      baud$none               ; parallell
```

```

0120' 47 44 50 20                db     'GDP   '             ; GDP-Ausgabetreiber
0133' 02                         db     mb$output            ; nur Ausgabe
0134' 00                         db     baud$none            ; paraliell (ueber BUS)

0135' 54 45 52 4D                db     'TERM  '             ; SER - Als Terminal-IO
013B' 0F                         db     mb$in$out+mb$serial+mb$soft$baud ; Ein/Aus mit umschalt.
013C' 0E              tbaud:     db     baud$9600            ; Baudrate

013D' 53 45 52 31                db     'SER1  '             ; SER - als AUX
0143' 0F                         db     mb$in$out+mb$serial+mb$soft$baud
0144' 0E              abaud:     db     baud$9600

0145' 53 50 52 49                db     'SPRINT'             ; SER - als Serialdrucker
014B' 0E                         db     mb$output+mb$serial+mb$soft$baud
014C' 0E              pbaud:     db     baud$9600

014D' 50 50 52 49                db     'PPRINT'             ; IO als CENTRONIC Schnittstelle
0153' 02                         db     mb$output
0154' 00                         db     baud$none

        0006          max$devices    equ    ($-@ctbl)/8

0155' 00                         db     0                    ; Tafelterminator

                                 end
0 Error(s) Detected. 342 Program Bytes.
174 Symbols Detected.
```

Cross Reference:

```
0047' ?CI              22   139
004D' ?CI1            152   153
0000' ?CINIT           22    35
0057' ?CIST            22   168
005D' ?CIST1          152   182
0065' ?CO              22   193
0069' ?CO1            203   204
0073' ?COST            22   219
0079' ?COST1          203   234
007F' ?COST2          160   185   211   237
0125' @CTBL            23    57   508   544
0144' ABAUD            91   534
000E  BAUD$9600       530   534   538
0000  BAUD$NONE       522   526   542
00E9' CI$TBL          159   426
00F5' CIST$TBL        184   439
0101' CO$TBL          210   452
010D' COST$TBL        236   465
008F' DUMMY           261   433   436   437   446   449   450   458   471
                      490   491   495
FC03  ECONIN          432
FC00  ECONIST         445
FC06  ECONOUT         459
0025' I$D2MSG          74    83
0033' I$D3MSG          93    96
0041' I$D4MSG         106   109
0014' I$DEV2           72   492
001B' I$DEV21          75    94   107
002A' I$DEV3           91   493
0038' I$DEV4          104   494
0046' I$DEV5          117
0023# INISUB           25    81
0049  IOCMD           357   359   414
0048  IODAT           355
0006  MAX$DEVICES      46   149   179   201   231   544
0003  MB$IN$OUT       529   533
0001  MB$INPUT        521
0002  MB$OUTPUT       525   537   541
0008  MB$SERIAL       529   533   537
0004  MB$SOFT$BAUD    529   533   537
0119' MY$DEF           51   478
014C' PBAUD           104   538
013C' TBAUD            24    72   530
00FE  UCMD             85
00EE  UCMD1           111
00CE  UCMD2            98
00FF  UCON             84
00EF  UCON1           110
00CF  UCON2            97
00FC  UDAT            259   317
00EC  UDAT1           342
00CC  UDAT2           271   329
00FD  USTA            287   384
00ED  USTA1           405
00CD  USTA2           299   396
0086' _CI2            251   258   434
0090' _CI3            263   270   435
009A' _CIST2          257   281   447
```

```
Cross Reference:

009C' _CIST21        288   300
00A2' _CIST3         269   293   448
00A6' _CO2           308   315   460
00AF' _CO3           320   327   461
00B8' _CO4           332   340   462
00C1' _CO5           345   353   463
00D0' _COST1         368   472
00D3' _COST2         314   378   473
00D5' _COST21        385   397   406
00DB' _COST3         326   390   474
00DF' _COST4         339   399   475
00E3' _COST5         352   408   476
0119' _DEV0          490
011B' _DEV1          491
011D' _DEV2          492
011F' _DEV3          493
0121' _DEV4          494
0123' _DEV5          495
```

Kapitel 8 Das BIOS-Segment MOVE

In diesem Programmsegment werden die Unterprogramme MOVE, XMOVE und die Bankumschaltung vorgenommen.

Das MOVE-Unterprogramm prüft vor jedem Speichertransfer nach, ob dieser Transfer innerhalb einer Bank oder innerhalb verschiedener Banks vorgenommen werden muß. Information hierüber gibt ein vom Unterprogramm XMOVE gesetztes Flag.

Der Datentransfer innerhalb verschiedener Banks kann nur über einen Puffer im gemeinsamen Speicherteil (Common-Area) vorgenommen werden. Es stehen hierzu zwei Möglichkeiten zur Verfügung:

1. Im SCB ist (vom BDOS vorgegeben) ein 128-Byte Puffer angegeben, der zu diesem Zweck verwendet werden kann. (BNKBF Offset 35H) Dieser Puffer wird auch vom BDOS für so manches verwendet, steht jedoch bei Aufruf von MOVE zur Verfügung.

2. Es wird ein Extra-Puffer im gemeinsamen Bereich deklariert, der praktisch von beliebiger Größe sein kann. Sinnvoll ist dieser Puffer, wenn er Datenblöcke von 256, 512 oder 1024 Bytes verwalten kann. Zwischengrößen bringen keinen Vorteil, da der Transfer meist in der Größe von ganzen Sektoren erfolgt.

Wird ein extra Puffer verwendet, so kann der Transfervorgang bei entsprechender Puffergröße erheblich gesteigert werden. Die Deklaration des Puffers kann als Reservierung im BIOS erfolgen oder wie im gezeigten Beispiel als Reservierung eines festen Speicherplatzes, der nicht unter die von GENCPM verwaltete Speicherbelegung fällt.

Die Bankumschaltung muß den gegebenen physikalischen Möglichkeiten entsprechen. Falls das Ausgabeport nicht rücklesbar ist, müssen entsprechende Vorkehrungen getroffen werden um jederzeit Information über die gerade gültige Bank erhalten zu können.

Das nachfolgende Listing zeigt ein entsprechendes Treiberprogramm:

```
                             MACLIB  DEFAULT.INC      ; Vereinbarungen
                     ;**************************************************************
                     ;*                                                            *
                     ;* rel 1.0 vom 07.01.85                                       *
                     ;*                                                            *
                     ;* Das MOVE-Modul ist fuer Banking und den Programmtransfer    *
                     ;* allgemein und 'zwischen' den Banks zustaendig              *
                     ;* Die Bankumschaltung erfolgt ueber Port BBPORT der Bank-    *
                     ;* Bootkarte.                                                 *
                     ;*                                                            *
                     ;* geschrieben von RAOUL O. KOERBER                           *
                     ;* nach Unterlagen von DRI                                    *
                     ;*                                                            *
                     ;**************************************************************

                             cseg                     ; gemeinsamer Bereich

                             ;
                             ; ****************
                             ; Systemadressen
                             ; ****************
                             ;

                             public ?move,?xmove,?bank
                             extrn   @cbnk

                             ;
                             ; ********************************
                             ; Programmstart
                             ; ********************************
                             ;


                             ;
                             ; ****************
                             ; Interbank-MOVE
                             ; ****************
                             ;

0000'                ?xmove:

                             ;
                             ; Aufruf mit <B>=Zielbank und <C>=Quellbank
                             ; Banken werden abgelegt und ein Flag gesetzt
                             ; damit bei naechstem ?MOVE Interbankmove veranlasst wird
                             ;

0000' ED 43 005D'            ld      (dsbnk),bc       ; Banknummern retten
0004' 3E FF                  ld      a,-1             ; und XMOVE-Flag
0006' 32 005C'               ld      (xmflg),a        ; setzten
0009' C9                     ret

                             ;
                             ; ****************
                             ; Hilfsprogramme
                             ; ****************
                             ;

000A'                xmovel:
```

```
                                ;
                                ; Einsprung von ?MOVE mit Ziel in <DE> und Quelle in <HL>
                                ; die Laenge des Transfers ist in <BC>
                                ; es ist ein Puffer unter dem Label A$BBUF eingerichtet mit
                                ; der Laenge A$BLEN
                                ; Puffer und Laenge koennen veraendert werden ( in DEFAULT.INC )
                                ; es ist dann aber unbedingt darauf zu achten das in GENCPM.DAT
                                ; ( oder direkt mit GENCPM ) das RAM-Ende (MEMTOP) ent-
                                ; sprechend angepasst wird.
                                ;

000A' E5                        push    hl              ; Zeiger auf Quellcode retten
000B' 21 0200                   ld      hl,a$blen       ; passt MOVE mit einem Transfer ?
000E' B7                        or      a               ; kein CARRY
000F' ED 42                     sbc     hl,bc           ; CARRY wenn MOVE zu gross !
0011' E1                        pop     hl              ; Zeiger auf Quellcode
0012' 38 0F                     jr      c,xmove2        ; mehrfacher Transfer ...
0014' CD 0031'                  call    xmove4          ; Ausfuehrung des Transfers...
0017' 3A 0000#                  ld      a,(@cbnk)       ; danach wieder gueltige Bank ...
001A' CD 0059'                  call    ?bank           ; ... selektieren
001D' AF                        xor     a               ; und XMOVE-Flag
001E' 32 005C'                  ld      (xmflg),a       ;   zuruecksetzen
0021' EB                        ex      de,hl           ; Zeiger in richtiger Reihenfolge
0022' C9                        ret

0023' 78               xmove2:  ld      a,b             ; Zaehler um A$BLEN vermindern
0024' D6 02                     sub     high a$blen
0026' 47                        ld      b,a
0027' C5                        push    bc              ; verbleibende Laenge retten
0028' CD 002E'                  call    xmove3          ; und Block transferieren
002B' C1                        pop     bc              ; ... verbleibende Laenge
002C' 18 DC                     jr      xmove1          ; bis fertig ...

002E' 01 0200          xmove3:  ld      bc,a$blen       ; Laenge des Transfer
0031' C5               xmove4:  push    bc              ; Laenge retten
0032' D5                        push    de              ; Zieladresse retten
0033' 3A 005D'                  ld      a,(dsbnk)       ; Quellbank
0036' CD 0059'                  call    ?bank
0039' 11 FA00                   ld      de,a$bbuf       ; Zwischenziel ist A$BBUF im COMMON-Bereich
003C' ED B0                     ldir                    ; MOVE !
003E' D1                        pop     de              ; ...Ziel
003F' C1                        pop     bc              ; ...Laenge
0040' E5                        push    hl              ; rette Anfang Quellcode des naechsten Blocks
0041' 21 FA00                   ld      hl,a$bbuf       ; Quelle ist jetzt A$BBUF
0044' 3A 005E'                  ld      a,(dsbnk+1)     ; Zielbank
0047' CD 0059'                  call    ?bank
004A' ED B0                     ldir                    ; MOVE !
004C' E1                        pop     hl              ; ... letze Quelladresse ( auf Bank )
004D' C9                        ret


                                ;
                                ; ****************
                                ; Standard-MOVE
                                ; ****************
                                ;

004E'                  ?move:
```

```


                                   ;
                                   ; Beim Einsprung ist in <HL> die Zieladresse (wohin)
                                   ; in <DE> die Quelladresse (woher) und in <BC> der Zaehler
                                   ; (wieviel).
                                   ; <HL> und <DE> muessen bei Rueckkehr auf die der Operation
                                   ; folgenden Speicherstellen Zeigen ( BC=0 )
                                   ;

004E' EB                           ex      de,hl           ; fuer Z80-Power HL/DE vertauschen
004F' 3A 005C'                     ld      a,(xmflg)       ; Test ob XMOVE
0052' B7                           or      a
0053' 20 B5                        jr      nz,xmovel       ; ... ja, XMOVE
0055' ED B0                        ldir                    ; sonst ungebankter MOVE
0057' EB                           ex      de,hl           ; Zeiger zurueck in 'alter' Ordnung
0058' C9                           ret

                                   ;
                                   ; ****************
                                   ; Bankumschaltung
                                   ; ****************
                                   ;

0059'                    ?bank:

                                   ;
                                   ; Bankselekt erfolgt ueber Port BBPORT auf Bank-Boot-karte
                                   ; die Banknummer wird in <A> uebergeben
                                   ;

0059' C3 FC24                      jp      esetbnk         ; Bank in ECURBNK ablegen und schalten

                                   ;
                                   ; ****************
                                   ; Variable
                                   ; ****************
                                   ;

005C' 00                 xmflg:    db      0               ; xmove aktiv wenn Flag = FF
005D' 0000               dsbnk:    dw      0               ; Ziel und Quellbank ( <B> - <C> )

                                   end
0 Error(s) Detected. 95 Program Bytes.
111 Symbols Detected.
```

Cross Reference:

```
0059' ?BANK        24   79   97  105  140
004E' ?MOVE        24  116
0000' ?XMOVE       24   39
0018# @CBNK        25   78
FA00  A$BBUF       98  103
0200  A$BLEN       72   86   93
005D' DSBNK        47   96  104  156
FC24  ESETBNK     147
005C' XMFLG        49   81  127  155
000A' XMOVE1       58   91  129
0023' XMOVE2       76   85
002E' XMOVE3       89   93
0031' XMOVE4       77   94
```

Kapitel 9 Das BIOS-Segment DISKIO

In diesem Programm-Modul werden alle Dinge ,die mit dem Zugriff auf Laufwerke und Disketten zu tun haben, bearbeitet.

Das Modul ist vom Prinzip her dreigeteilt:

1. Es enthält die Laufwerkstabelle DTBL.

   In dieser Tabelle sind die Adressen aller vorhandenen DPH's zusammengefaßt. Unbelegte Laufwerke enthalten 0000H als Vektor.

2. Es enthält die XDPH's aller belegter Laufwerke

   mit Zeiger auf die zugehörigen DPB's und die SKEW-Translate Tabelle. Außerdem enthält jeder XDPH (eXtendet - erweiterter DPH) Zeiger auf die laufwerkstypische Schreib-, Lese-,Init- und Loginroutine.

3. Es enthält unter Punkt 2 genannte Unterprogramme und dazugehörende Hilfsprogramme.

Zum besseren Verständis der in DISKIO vorhandenen Unterprogramme sind einige Erläuterungen unumgänglich:

Das BDOS greift auf die Disketten mit einer Folge von Aufrufen unterschiedlicher BIOS-Funktionen (in die Sprungtabelle) zu.

Diese Funktionen veranlassen der Reihe nach:

Ansprechen des entsprechenden Laufwerkes	SELDSK
Ausgeben der logischen Sektornummern	MULTIO
Setzen der Track-Nummer	SETTRK
Setzen der Sektor-Nummer	SETSEC
Setzen der DMA Adresse	SETDMA
Setzen der Bank (bei gebanktem System)	SETBNK

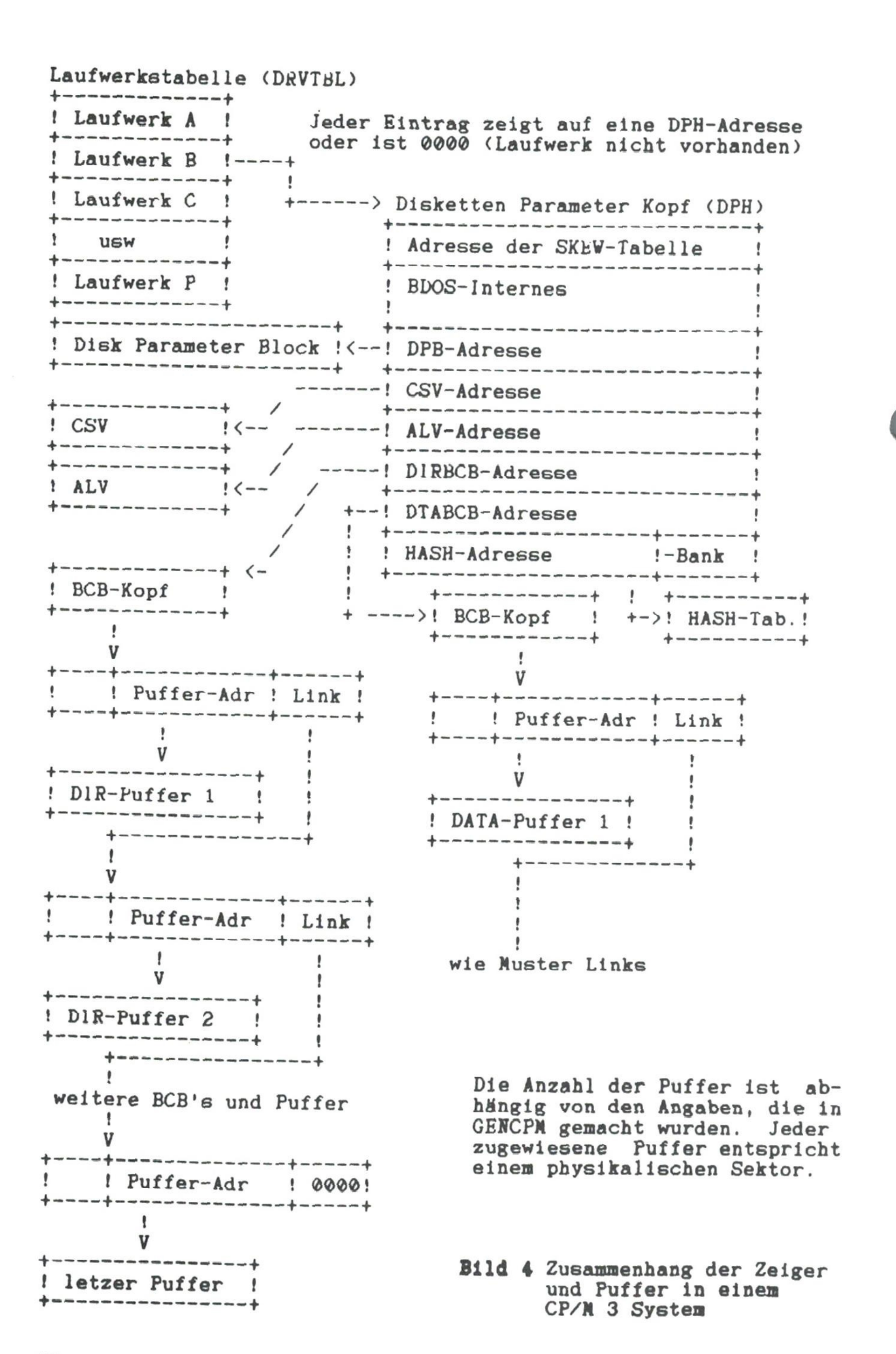
nachdem diese Parameter festgelegt sind folgt der

Lesezugriff	READ
oder	
Schreibzugriff	WRITE

Letzteres sind die einzigen physikalischen Zugriffe auf das Laufwerk; nur beim Lese- oder Schreibzugriff wird auch das Laufwerk selektiert und, falls notwendig die richtige Spur (Track) angesprochen.

Diese Sequenz wird bei SETTRK solange wiederholt, bis der entsprechende READ/WRITE Vorgang abgeschlossen ist. Bei Laufwerkswechsel beginnt der Vorgang wieder entsprechend bei SELDSK.

Die Details über Laufwerksdaten, Tracks, Sektoren usw holt sich das BDOS aus unterschiedlichen Quellen die in der nachfolgenden Skizze zum besseren Verständnis zusammenhängend aufgezeigt werden sollen:



Die Anzahl der Puffer ist abhängig von den Angaben, die in GENCPM gemacht wurden. Jeder zugewiesene Puffer entspricht einem physikalischen Sektor.

Bild 4 Zusammenhang der Zeiger und Puffer in einem CP/M 3 System

Aus Bild 4 ist zu entnehmen, daß zuerst die Adresse des DPH's aus der Laufwerkstabelle entnommen wird.

Aus diesem Datenblock werden nun die Zeiger auf die unterschiedlichsten weiteren Zeiger übernommen:

1. Hier findet BIOSKRNL die Adresse der INIT, LOGIN, LESE- und SCHREIB- Unterprogramme zu dem angesprochenen Laufwerk.

2. Hier findet das BDOS alle (technischen) Angaben des Laufwerkes im DPB.

3. Hier sind die Adressen der BCB's (Puffer Kontroll Block) für die Datenpuffer und den Puffer des Inhaltsverzeichnisses.

   Die Angelegenheit mit den BCB's ist sicherlich etwas verwirrend. Hier sind gleich mehrere Puffer, jeder mit einem eigenen BCB eingetragen. Das BDOS findet den jeweils nächsten Puffer anhand einer sog. LINK-Adresse, sie ist es, die auf den nächsten BCB zeigt.

   Ist die Linkadresse 0000, so bedeutet dies, daß der letzte Puffer gefunden wurde.

4. Hier ist die Adresse der HASH-Tabelle zu finden.

Übrigens überträgt das BDOS bei jeder Erstanspache eines Laufwerkes (loggin) alle Daten aus dem DPH und dem DPB in BDOS-eigene Puffer, um schneller auf diese Daten zugreifen zu können.

Hierin ist auch der Grund zu suchen, warum es nicht erlaubt ist, beim Erkennen eines Diskettenwechsels (und Umschalten des Formates) innerhalb der Schreib- Leseroutine, einfach den DPH abzuändern, sondern, daß in diesem Falle ein entsprechender Fehler gemeldet werden muß, denn nur in diesem Falle liest das BDOS neue XDPH und DPB -Daten ein.

Nun das ganze noch einmal im Detail:

Welches Laufwerk angesprochen werden soll, muß dem BDOS vom Benutzer (in der CCP-Befehlszeile) bzw. vom Programm mitgeteilt werden.

Mit dieser Information wird die BIOS-Funktion SELDSK aufgerufen. Diese Funktion entnimmt aus der Laufwerkstabelle DRVTBL die Adresse des zum Laufwerk gehörenden XDPH (erweiterten Disk-Parameter-Header).

Diese Adresse wird dem BIOS übergeben - und damit alle Information über das entsprechende Laufwerk.

Wird das Laufwerk erstmals angesprochen, wird von SELDSK ein spezielles Erkennen der Diskette angefordert (loggin), wobei der entsprechende XDPH an das evtl. neue Diskettenformat angepaßt werden kann.

Der Aufbau eines eXtended Disk Parameter Headers sieht wie folgt aus:

```
Adresse          LOW BYTE          HIGH BYTE

                 0               7 8              15
                 +----------------+----------------+
XDPH-10          !      Adresse von WRITE          !
                 +---------------------------------+
XDPH-8           !      Adresse von READ           !
                 +---------------------------------+
XDPH-6           !      Adresse von LOGIN          !
                 +---------------------------------+
XDPH-4           !      Adresse von INIT           !
                 +----------------+----------------+
XDPH-2           !     UNIT       !   TYP          !
                 +----------------+----------------+
XXXDPHXX         ! Adresse des TRANSLATE Tabelle   !
                 +----------------+----------------+
XDPH+2           !     0    CP/M Scatch     0      !
                 +----------------+----------------+
XDPH+4           !     0          !     0          !
                 +----------------+----------------+
XDPH+6           !     0          !     0          !
                 +----------------+----------------+
XDPH+8           !     0          !     0          !
                 +----------------+----------------+
XDPH+10          !     0          !  Media-Flag    !
                 +----------------+----------------+
XDPH+12          !       Adresse des DPB           !
                 +---------------------------------+
XDPH+14          !       Adresse des CSV           !
                 +---------------------------------+
XDPH+16          !       Adresse des ALV           !
                 +---------------------------------+
XDPH+18          !      Adresse des DIRBCB         !
                 +---------------------------------+
XDPH+20          !      Adresse des DTABCB         !
                 +---------------------------------+
XDPH+22          !   Adresse der HASH-Tabelle      !
                 +----------------+----------------+
XDPH+24          !   HASH-Bank    !
                 +----------------+
```

Bild 5 Aufbau des XDPH's

Die Zeiger auf WRITE, READ, LOGIN und INIT sind nur für das BIOS von Interesse, BDOS greift nicht darauf zu.

Prinzipiell könnten diese Vektoren entfallen - sie machen aber den Zugriff auf die entsprechenden Funktionen etwas einfacher - da von BDOS immer die Adresse von XXXXDPHXXX des entsprechenden Laufwerkes übergeben wird.

Die Zeiger UNIT und TYPE wurden im nachfolgenden Beispiel etwas 'missbraucht'. Unter UNIT ist die zum Laufwerk passende STEP-Rate (als STEP-HOME-Wert) abgelegt und unter TYPE ist der zum Laufwerk gehörende SELECT-Code abgelegt. Dies vereinfacht die Anpassung von Laufwerken unterschiedlicher Charakteristik erheblich. Das BDOS selbst benötigt nur den 'Rest' der Tabelle, beginnend

mit der Adresse der

TRANSLATE oder kurz SKEW-Tabelle.
Um schneller auf die Disketten zugreifen zu können, werden die Sektoren eines Tracks meist nicht hintereinander geschrieben, sondern (mit einem sogenannten SKEW-Faktor), versetzt z.B. nach Sektor 1 wird Sektor 7, dann Sektor 13 usw geschrieben. Wie dieser SKEW-Faktor aussieht, läßt sich aus der XLT-Tabelle errechnen - den Vektor darauf findet CP/M bei XDPH+0, diesen Vektor übergibt das BDOS an die BIOS-Funktion SECTRN die ihrerseits die Nummer des entsprechenden physikalischen Sektors errechnet und zurückgibt.

9 Bytes werden vom BDOS für unterschiedliche Vektoren benutzt.

MEDIAFLAG dieses Flag wird vom BDOS auf 0 gesetzt, wenn ein Laufwerk ein'geloggt' wurde.

Das BIOS kann dieses Flag auf FF setzen, wenn es die (physikalische) Möglichkeit hat zu Erkennen ob ein Laufwerkschacht geöffnet wurde oder nicht. Das BDOS prüft vor jedem Diskettenzugriff dieses Flag, ist es gesetzt, wird die Prüfsumme des Inhaltsverzeichnisses mit dem letzten Eintrag verglichen, wird eine Differenz gefunden, wird SELDSK aufgerufen mit Bit 0 des Registers E auf 1 gesetzt.

DPB zeigt auf den sog. Disk-Parameter-Block, dessen 'Inhalt' folgende Bedeutung hat:

FELD	BITS	Bedeutung
SPT	16b	SPT gibt die Anzahl der Sektoren pro Track an und zwar in logischen 128 Byte Sektoren.
BSH	8b	BSH ist der Block-Shift-Faktor abhängig von der Blockgröße 1-2-4-8-16k = 3-4-5-6-
BLM	8b	BLM ist die Blockmaske wie BSH abhängig von der Blockgröße 1-2-4-8-16k = 7-15-31-63-127
EXM	8b	EXM =Extend Maske abhängig von DSM und der Blockgröße. Bei DSM < 256 = 0-1-3-7-15 und DSM > 255 = X-0-1-3-7. (X=nicht möglich). (Blockgrößen 1-2-4-8-16K)
DSM	16b	DSM gibt die Anzahl von Blöcken auf einer Disk an (-1). Die Gesamtkapazität errechnet sich mit Blockgröße X (DSM+1) - ohne Systemtracks
DRM	16b	Anzahl der (möglichen) Einträge im Inhalts-Verzeichnis (-1)
AL0	8b	AL0 Blockreservierung - jedes Bit beginnend
AL1	8b	mit BIT 7 von AL0 und endend mit BIT 0 von AL1 bedeutet gesetzt, daß der entsprechende Block belegt ist, entsprechend falls nicht gesetzt (=0), daß der Block ist frei (unbelegt(ist
CKS	16b	CKS (DRM/4+1) gibt an wieviel (log) Sektoren zum Berechnen der Prüfsumme des Inhaltsverzeichnisses gelesen werden müssen (wichtig zum Erkennen eines Diskettenwechsels. Für nichtwechselnde Platten (Harddisk) kann CKS=8000h gesetzt werden

FELD	BITS	Bedeutung
OFF	16b	Gibt an wieviel Tracks (für CPMLDR) reserviert sind. Die Directory beginnt sofort danach
PSH	8b	PSH physikalischer record shift factor
PHM	8b	PHM physikalische record maske
		Dies sind Faktoren und Masken, die von der Blockgröße wie folgt abhängen:
		Block 1-2-4-8-16k = PSH 0-1-2-3-4-5
		PHM 0-1-3-7-15-31

CP/M bezieht aus den Angaben des DPB's alle Informationen die es über die jeweils angesprochene Diskette benötigt, wie Start des Inhaltsverzeichnis, freie Blöcke, Größe und Anzahl der physikalischen Sektoren pro Track usw.

Mit dieser Hilfe kann es auf das Inhaltsverzeichnis (DIRectory) zugreifen, in welchem wiederum alle Informationen stehen (oder von CP/M geschrieben werden), die Auskunft darüber geben, auf welchem Track und in welchem Sektor ein bestimmter Programmteil steht oder geschrieben werden kann.

CSV ist die Adresse eines Feldes, aus dem eine diskettentypische Prüfsumme errechnet werden kann. Die Größe des Feldes entspricht DRM/4+1 (=CKS im DPB).

ALV ist die Adresse eines Felde, wo BDOS die Belegung (Allocation) der Diskette ablegt. Die Größe des Feldes entspricht DSM/4+2.

DIRBCB ist die Adresse eines Buffer-Controll-Blocks für das Inhaltsverzeichnis. In diesem BCB können mehrere Zeiger auf Pufferadressen abgelegt sein, in welchen das Inhaltsverzeichnis zwischengespeichert wird.

DTABCB ist die Adresse eines Buffer-Controll-Blocks für ein Datenfeld, welches als Datenpuffer beim Schreiben und Lesen von Diskette dient.

HASH ist die Adresse eines speziellen (optionalen) Inhaltsverzeichnises das über einen speziellen Algorithmus sofort den Sektor errechnet, in welchem die gesuchte Datei sein kann. Da damit das langwierige Durchforsten des gesamten Inhaltsverzeichnisses drastisch verkürzt werden kann, bringt diese Option viel Gewinn beim Diskettenzugriff - benötigt andernseits aber entsprechenden Speicherplatz. Offensichtlich wird der Geschwindigkeitsgewinn erst bei Inhaltsverzeichnissen mit mehr als 128 Einträgen.

HBANK gibt an auf welcher Bank sich das HASH-Inhaltsverzeichnis befindet.

Alle Angaben zu CSV bis HASH werden, falls 0FFFEH in die Tabelle eingetragen wird, von GENCPM.COM angepaßt. HBANK wird ebenfalls gesetzt. Wird HASH nicht zugelassen, ist der Vektor auf 0FFFFH zu setzen.

Die Angaben, die zum Aufbau des DPB (Disk-Parameter-Block) gemacht wurden, bedürfen sicherlich noch einer ausführlicheren Behandlung.

Allem voran ist ein immer wiederkehrender Begriff zu erläutern, die Blockgröße.

Es handelt sich dabei um die Datenmenge die als kleinste Einheit einen Eintrag im Inhaltsverzeichnis belegt. Diese Blockgröße kann 1024, 2048, 4096, 8192 oder 16384 Bytes belegen.

Eine Blockgröße von 1024 Bytes kann nur auf Disketten mit einer max. Kapazität von 255K verwendet werden. Dies hängt damit zusammen, daß hier nur Blocknummern von 00...FFH (0...255) verwendet werden. Bei Blockgrößen über 1024 Bytes werden Blocknummern von 0000...7FFF verwendet. Die entspricht einer maximal ansprechbaren Diskettenkapazität von 512MB.

Jeder Eintrag im Inhaltsverzeichnis belegt nun im Mindestfalle einen Block, im Maximalfalle belegt er 16 Blöcke (Wenn eine Datei größer als 16K wird, werden bis zu 31 zusätzliche Einträge im Inhaltsverzeichnis vorgenommen, beim DIR-Befehl wird jedoch immer nur der Namen des ersten Eintrages ausgegeben). Dies bedeutet jedoch auch, daß eine Datei nicht größer als 512K werden kann - eine ganze Menge -.

Aus dem oben Gesagten kann aber auch erkannt werden, daß mit einer Blockgröße von 2K und einer vorgegebenen Anzahl von 64 Einträgen im Inhaltsverzeichnis minimal 64X2=128K und maximal 64X16=1024K belegt werden kann.

Geht man von einer mittleren Dateigröße von 10K aus, so bedeutet dies, daß mit 64 möglichen Einträgen nur 640K pro Diskette abgelegt werden können - gleichgültig ob 800K oder 1200K paßen würden, und sind es ganz kleine Dateien, würde sich die Zahl auf 128k verringern.

Um dieses Problem zu umgehen, sind zwei Methoden möglich:

1. Die Anzahl der möglichen Einträge im Inhaltsverzeichnis wird erhöht.

   Bis 1024 Einträge ist dies ein gangbarer Weg, darüber kann es problematisch werden für den Fall, daß 'Rettungsprogramme' für versehentlich gelöschte Dateien benötigt werden.

2. Die Blockgröße wird verdoppelt (bis 16K-Blöcke).

Bei der Zuordnung von Blockgröße und Anzahl der Einträge in das Inhaltsverzeichnis sind immer mehrere Faktoren zu berücksichtigen:

werden überwiegend kleine Dateien (bis 32K) bearbeitet, ist eine Blockgröße von 2K sehr sinnvoll. Sind die Mehrzahl der Dateien über 32K groß oder die Gesamtkapazität über 16MB, ist eine Blockgröße von 4K zu empfehlen.

Die Anzahl der Einträge in das Inhaltsverzeichnis hängt in erster Linie von der Gesamtkapazität der Diskette ab. Ein recht guter Richtwert ist es, die mittlere Dateigröe auf 8K zu schätzen, bei einer Diskettenkapazität von 800K ergäbe

dies ein Inhaltsverzeichnis von rd. 100 Einträgen. Da bei Computern am besten mit Binärwerten gerechnet wird, ergibt sich ein Wert von 128 (2^7) Einträgen.

In der Praxis steht man hier vor einem ganz anderen Problem, man möchte u.U. kompatibel zum Format eines Anderen sein, den leider hat hier Digital Research keine Normung vorgegeben - so 'wurstelt' halt jeder nach seinem Geschmack - mit teilweise recht 'lustigen' Kombinationen.

Nun zu den Einträgen im DPB im Einzelnen:

SPT Gibt die Gesamtzahl der (logischen 128-Bytes) Sektoren pro Track an.

Track kann hier sehr unterschiedlich gedeutet sein, falls es sich um eine zweiseitige Diskette handelt.

Zweiseitige Disketten können prinzipiell mit zwei unterschiedlichen Methoden von Seite zu Seite umgeschaltet werden.

Eine Methode (die im Beispiel angewandte) besteht darin,die Tracks auf der Vorderseite und der Rückseite getrennt zu zählen, eine Diskette mit 40 Track hätte demnach 40 Tracks auf der Vorderseite und 40 Tracks auf der Rückseite, also (logisch) 80 Tracks. In diesem Falle ist SPT die Anzahl der Sektoren je Track und Seite.

Die zweite Möglichkeit ist es, die Seitenumschaltung in Abhängigkeit von der Sektoranzahl pro Seite zu machen. Eine zweiseitige Diskette mit z.B. 5 Sektoren pro Seite und Track würde in diesem Falle für SPT die Angabe 10 Sektoren pro Track bedeuten, bei 40 Tracks gesammt.

BSH aus diesem Wert errechnet sich das BIOS die Blockgröße. Es gilt folgender Zusammenhang:

BLS=128^BSH

D.h. Die Blockgröße (BLS) ist die BSH'te Potenz von 128Bytes (der logischen Sektorgröße)

BLM gibt die Anzahl der logischen Sektoren in einem Datenblock an, mit der Besonderheit, daß der Zähler mit 0 beginnt.

Entsprechend ist BLM für die Blockkgröße 1024=7 und die Blockgröße 2048=15. Allgemein gilt

BLM=(BLS/128)-1

EXM Auch dieser Wert ist direkt abhängig von der Blockgröße, darüberhinaus auch noch von der Gesamtkapazität einer Diskette.

Aus EXM läßt sich direkt die Blockgröße als ein Vielfaches der kleinstmöglichen Blockgröße errechnen.

Es gilt folgender Zusammenhang:

BLS	EXM - Werte	
	DSM<256	DSM>246
1024	0	unerlaubt
2048	1	0
4096	3	1
8192	7	3
16384	15	7

DSM Dieser Wert enthält Angaben über die gesamte Speicherkapazität einer Diskette.

Er errechnet sich nach der Formel:

DSM=((SPT*128*Tracks)/BLS)-1

Bei Laufwerken mit einer Kapazität bis 256K, darf DSM maximal 00FFH groß sein, bei größeren Kapazitäten max. 7FFFH.

DRM Diese Angabe steht für die maximal mögliche Anzahl von Einträgen im Inhaltsverzeichnis (Einträge -1).

Das Inhaltsverzeichnis benötigt für jeden Eintrag 32 Bytes. Der Wert von DRM muß gleich oder kleiner dem Wert von

(BLS/32*16)-1

sein.

Bei einer Blockgröße von 2048 Bytes sind also maximal 1024 Einträge im Inhaltsverzeichnis zuläßig.

AL0
AL1 Dieses 16 Bit-Wort enthält je gesetzem Bit (Bit=1) beginnend mit Bit 15 - AL0, Angaben über die Anzahl der vom Inhaltsverzeichnis belegten Blöcke.

Sind z.B. 128 Einträge bei einer Blockgröße von 2048K vorgesehen, wird der Wert

```
128*32 (Einträge*Eintragsgröße)
------                              = 2
2048   dividiert durch Blockgröße
```

Der Belegungsvektor sähe also wie folgt aus:

1100000000000000 = C000H

OFF gibt die Anzahl der (für den CPM-Lader) reservierten Track, beginnend mit Track 0 an.

OFF entspricht dem Track,in welchem das Inhaltsverzeichnis beginnt.

Einen weiteren Aspekt bietet die Angabe von OFF noch: bei Laufwerken mit hoher Kapazität (z.B. Harddisks) kann durch OFF ein Laufwerk in mehrere Segmente eingeteilt werden und so besser überschaubare kleinere Einheiten 'geschaffen' werden.

PSH gibt Angaben zur physikalischen Sektorgröße in der Form

Sektorgröße= 128^(PSH+)

PHM gibt ebenfalls Angaben zur phys. Sektorgröße in der Form

Sektorgröße=128X(PHM+1)

Bleibt noch ein Wort zum Aufbau des BCB's, des Puffer-Kontroll-Blockes, auf welchen DIRBCB und DTABCB im DPH zeigt.

Alle BCB's sind gleich aufgebaut in der Form:

FELD	BITS	Bedeutung
DRV	8	Laufwerksnummer in HEX (FF wenn die Zuweisung nicht über GENCPM erfolgt) welchem der Block zugewiesen ist
REC#	24	Angaben über die Position innerhalb des ganzen Datenfeldes auf welche die Adresse in BUFFAD zeigt. REC# gibt Absolute Sektornummer beginnend mit 000000H an.
WFLG	8	Wird vom BDOS auf FF gesetzt, wenn er Daten enthält,die noch nicht bearbeitet sind. Sind die Daten auf Diskette geschrieben wird das Byte=00 gesetzt.
00	8	Benutzt vom BDOS
TRACK	16	Tracknummer aus welchem in den Puffer gelesen (oder geschrieben) wird.
SECTOR	16	physikalische Sektornummer
BUFFAD	16	Adresse des Puffers in welchem die Daten abgelegt sind.
BANK	8	Bank in welcher sich der Puffer befindet.
LINK	16	Adresse des nächsten BCB's mit Angaben zum nächsten Puffer. Ist die LINK-Adresse=0000H bedeutet dies, daß keine weiteren Puffer zugewiesen wurden.

Dies alles klang nun nicht nur recht kompliziert, sicherlich ist es das auch. Aber alle Angaben, die hier gemacht wurden, sind eigentlich nur für den Spezialisten, der 'normale' Benutzer hat so gut wie nichts damit zu tun, bis auf wenige Ausnahmen, nämlich dann, wenn neue Laufwerke und/oder neue Diskettenformate in das vorhandene Betriebssystem eingebunden werden sollen.

Zur Vereinfachung für den Benutzer sind alle hierzu notwendigen Angaben für XDPH,DPB und die SKEW-Tabellen mit MACROS aufgebaut. Die Macros befinden sich in der Datei CPM3.INC. (siehe hierzu Kapitels 10 - Zuweisungen - Vereinbarungen - Macros).

Die ganze 'Arbeit' besteht dann nur noch im Eintragen der passenden Daten.

Das nachfolgende Listing dürfte sicherlich der 'komplizierteste' Teil im CP/M 3 BIOS sein.

Hier ist ein weites Feld für den Hobby- und den Systemprogrammierer gegeben. Leider führen die vielen Möglichkeiten aber auch zu Irrwegen und vor allem zur Inkompatibilität der Diskettenformate. Da Digital Research nur ein einziges Diskettenformat genormt hat, fühlte sich jeder Hersteller eines Computers nahezu verpflichtet ein unkompatibles Diskettenformat zu wählen, meist mit dem Hintergedanken des Datenschutzes und der vordergründigen Behauptung ein besonders effektives Format zu benutzen.

Das nachfolgende Listing zeigt als Beispiel einen Treiber für das Diskettenformat auf MINI-Laufwerken 2-seitig mit doppelter Dichte und 80 Tracks pro Seite. Alle diesen Daten waren bei der Implementierung vorgegeben, da die Disketten kompatibel zu vorhandenen CP/M 2 Systemen bleiben sollte.

Zur Seitenumschaltung wurde das verbreitetste Verfahren verwendet (nicht notwendigerweise auch das Schnellste).

Die Diskette ist (logisch) in 2X80 Tracks (=160) Tracks aufgeteilt, wobei sich physikalisch jeder gerade Track (0,2,4,..) auf Seite 0 der Diskette und jeder ungerade Track (1,3,5,...) auf Seite 1 der Diskette befindet. Von dieser Trackumschaltung abgesehen, 'sieht' das BIOS auf jedem Track dann 5 Sektoren zu je 1024 Bytes.

Der Algorithmus der Seitenumschaltung ist besonders einfach: Tracknummer geteilt durch 2 ist die neue (physikalische) Tracknummer. Verblieb bei der Division ein Rest handelt es sich anschließend um Sektoren auf Seite 1, andernfalls um Sektoren auf Seite 0.

Die Sektorgröße wurde mit 1024 Bytes pro Sektor vorgegeben. Diese 'großen' Sektoren sind unter CP/M 3 günstig, da mit einem Diskettenzugriff (ohne besondere Vorkehrungen) ein Kilobyte Daten gelesen oder geschrieben werden kann. Entgegen der weitverbreiteten Annahme ist dies der schnellstmögliche Diskettenzugriff. Sicher werden kleinere Sektoren 'schneller' gelesen, aber der Zugriff auf einen Sektor, bis er berechnet und gefunden ist dauert immer länger als das reine Lesen oder Schreiben.

Das Listing hat einen Treiber für den weitverbreiteten Laufwerks-Kontroller WD179x.

Er wird im Beispiel ohne Interrupt im reinen POLL-Modus betrieben. Damit ist es in einem 4-MHz-System zwar möglich MINI-Laufwerke mit doppelter Dichte zu betreiben, aber 8 Zoll-Laufwerke nur in einfacher Dichte - ausreichend zum Lesen und Schreiben im Standard-CP/M-Format.

Bei 6-MHz-Systemen (wenn auch bedingt) oder mit einer DMA kann selbstverständlich auch bei 8"-Laufwerken doppelte Schreibdichte erreicht werden.

```
                         MACLIB  DEFAULT.INC     ; Vereinbarungen
                         MACLIB  CPM3.INC        ; Macros
                   ;*************************************************************
                   ;*                                                           *
                   ;* rel 1.0 V003        850817                                *
                   ;*                                                           *
                   ;* Dieser Disketten-Treiber ist geschrieben fuer die         *
                   ;* NDR-FL02 mit dem kontrollierbaustein 1793                 *
                   ;*                                                           *
                   ;* geschrieben von RAOUL O. KOERBER                          *
                   ;* nach Unterlagen von GRI                                   *
                   ;*                                                           *
                   ;*************************************************************

                         ;
                         ; Variable Diskettenparameter und Bankroutinen
                         ;

                         extrn   @drv,?bank
                         extrn   @dma.@trk.@sect
                         extrn   @dbnk,@cbnk
                         extrn   ipchl

                         ;
                         ; Fehlerausgabe - Unterdrueckung
                         ;

                         extrn   @ermde

                         ;
                         ; allgemeine BIOS-Unterprogramme
                         ;

                         extrn   ?wboot,?pmsg,?pdec,?pderr,?conin,?cono
                         extrn   ?const

                         ;
                         ; Laufwerks-Motorabschaltung
                         ;

                         global  motoff,@dtbl

                         dseg                            ; Bank 0

                         ;
                         ; ********************************
                         ; Laufwerks-Parameter
                         ; ********************************
                         ;
                         ; ***************
                         ; Laufwerkstabelle
                         ; ***************
                         ;

 0000" 002A"       @dtbl:  dw      fda
 0002" 004D"               dw      fdb
 0004" 0000                dw      fdc
 0006" 0000                dw      fdd
```

```
 0008" 0000                     dw      fde
 000A" 0000                     dw      fdf
 000C" 0000                     dw      fdg
 000E" 0000                     dw      fdh
 0010" 0000                     dw      fdi
 0012" 0000                     dw      fdj
 0014" 0000                     dw      fdk
 0016" 0000                     dw      fdl
 0018" 0000                     dw      fdm
 001A" 0000                     dw      fdn
 001C" 0000                     dw      fdo
 001E" 0000                     dw      fdp

                                ;
                                ; ****************
                                ; XDPH's
                                ; ****************
                                ;
                                ; 8-Zoll Laufwerke sind nur in single-density betreibbar
                                ;

 0020" 0075"                    dw      fd$write
 0022" 006D"                    dw      fd$read
 0024" 006C"                    dw      fd$login
 0026" 006B"                    dw      fd$init
 0028" 00                       db      00b                     ; home mit 3ms step
 0029" 21                       db      sela+minis              ; Selekt 5" double-density
 002A"                fda::     dph     xlt5,dpb80d
 002A" 0066"          A         defw    xlt5                    ; Adresse der SKEW-Tabelle (XLT)
 002C" 00 00 00 00A             defb    0,0,0,0,0,0,0,0,0       ; BDOS Bereich
 0035" FF             A         defb    -1                      ; Media-flag
 0036" 0000'          A         defw    dpb80d                  ; Adresse des DPB
 0038" FFFE           A         defw    -2                      ; CSV wird von GENCPM eingetragen
 003A" FFFE           A         defw    -2                      ; ALV wird von GENCPM eingetragen
 003C" FFFE   FFFE    A         defw    -2,-2,-2                ; DIRBCB,DTABCB,HASH und HASH-Bank
 0042" 00             A         defb    0                       ; wird von GENCPM eingetragen

                                ;
                                ; Laufwerk 'B'
                                ;

 0043" 0075"                    dw      fd$write
 0045" 006D"                    dw      fd$read
 0047" 006C"                    dw      fd$login
 0049" 006B"                    dw      fd$init
 004B" 00                       db      00b                     ; home mit 3ms step
 004C" 22                       db      selb+minis              ; Selekt 5" double-density
 004D"                fdb:      dph     xlt5,dpb80d
 004D" 0066"          A         defw    xlt5                    ; Adresse der SKEW-Tabelle (XLT)
 004F" 00 00 00 00A             defb    0,0,0,0,0,0,0,0,0       ; BDOS Bereich
 0058" FF             A         defb    -1                      ; Media-flag
 0059" 0000'          A         defw    dpb80d                  ; Adresse des DPB
 005B" FFFE           A         defw    -2                      ; CSV wird von GENCPM eingetragen
 005D" FFFE           A         defw    -2                      ; ALV wird von GENCPM eingetragen
 005F" FFFE   FFFE    A         defw    -2,-2,-2                ; DIRBCB,DTABCB,HASH und HASH-Bank
 0065" 00             A         defb    0                       ; wird von GENCPM eingetragen
```

```
                                   ;
                                   ; vorlaeufig keine weiteren Laufwerke ...
                                   ;

          0000          fdc        defl    nein
          0000          fdd        defl    nein
          0000          fde        defl    nein
          0000          fdf        defl    nein
          0000          fdg        defl    nein
          0000          fdh        defl    nein
          0000          fdi        defl    nein
          0000          fdj        defl    nein
          0000          fdk        defl    nein
          0000          fdl        defl    nein
          0000          fdm        defl    nein
          0000          fdn        defl    nein
          0000          fdo        defl    nein
          0000          fdp        defl    nein

                                   cseg                     ; im gemeinsamen Bereich

                                   ;
                                   ; ****************
                                   ; DPB's
                                   ; ****************
                                   ;
                                   ;
                                   ; dpb => [Sektor] Groesse,Anzahl,
                                   ;          [Track] anzahl,
                                   ;          Blockgroesse,
                                   ;          Direintraege
                                   ;    und reservierte Tracks fuer Systemspur(en)
                                   ;

                        ;dpb8s:         dpb     128,26,77,1024,64,2     ; 8" sssd

 0000'                  dpb80d: dpb     1024,5,160,2048,256,4   ; 5" 80Track 2seitig
 0000' 0028       A             defw    ??0005                  ; 128 Byte (log)-Sektoren pro Track
 0002' 04 0F      A             defb    ??0006,??0007           ; Block-shift und Maske
 0004' 00         A             defb    ??0008                  ; Extent-Mmaske
 0005' 0185       A             defw    ??0009                  ; Maximale Block-Anzahl
 0007' 00FF       A             defw    ??000A                  ; Maximale DIR-Eintraege
 0009' F0 00      A             defb    ??000B,??000C           ; Belegungs-Vektoren
 000B' 0040       A             defw    ??000D                  ; Pruefsumme
 000D' 0004       A             defw    4                       ; Reservierte Tracks (System)
 000F' 03 07      A             defb    ??000E,??000F           ; (phys)-Sektor Groesse- &  Shift-Maske

                                   ;
                                   ; 40 Track Format
                                   ;

                        ;dpb40d: dpb    1024,5,80,2048,128,1    ; 5" 40Track 2seitig
                        ;dpb40s: dpb    1024,5,40,1024,64,3     ; 5" 40Track 1seitig ELZET

                                dseg                            ; gebankt

                                   ;
                                   ; ****************
                                   ; Skew Faktoren
```

```
                          ; ***************
                          ;

               ;xltm:   skew    26,6,1              ; single density Standart-CPM

0066"          xlt5:    skew    5,1,1               ; double-density 5-Zoll (1k Sektor)
0066" 01       B        defb    ?nxtsec+1
0067" 02       B        defb    ?nxtsec+1
0068" 03       B        defb    ?nxtsec+1
0069" 04       B        defb    ?nxtsec+1
006A" 05       B        defb    ?nxtsec+1

                          ;
                          ; ******************************
                          ; via XDPH aufgerufene Programme
                          ; ******************************
                          ;

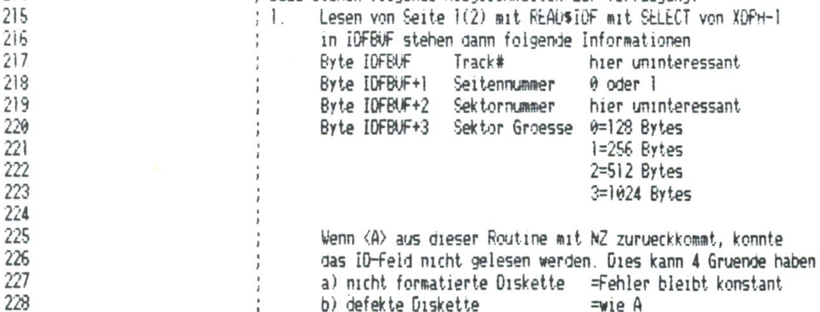
006B"          fd$init:

                          ;
                          ; wird beim Kaltstart angesprungen um evtl hardware-
                          ; Initialisierung des Floppy-Kontrollers vornehmen zu
                          ; koennen.
                          ; bei FL02 bleibt nichts zu tun
                          ;

006B" C9                  ret

006C"          fd$login:

                          ;
                          ; dieser Programmteil wird immer dann angesprungen wenn CPM
                          ; eine neue Diskette in einem Laufwerk vermutet.
                          ; Beim Ansprung zeigt <DE> auf den zugehoerenden XDPH
                          ; Notwendig ist dieser Programmteil bei moeglichem Formatwechsel
                          ; im gleichen Laufwerk. (Dichte-Sektorgroesse-Seitenwechsel)
                          ;
```



```
                          ; Es kann hier eine gewisse Art der Formaterkennung eingebaut werden
                          ; Dazu stehen folgende Moeglichkeiten zur Verfuegung:
                          ; 1.    Lesen von Seite 1(2) mit READ$IDF mit SELECT von XDPH-1
                          ;       in IDFBUF stehen dann folgende Informationen
                          ;       Byte IDFBUF     Track#          hier uninteressant
                          ;       Byte IDFBUF+1   Seitennummer    0 oder 1
                          ;       Byte IDFBUF+2   Sektornummer    hier uninteressant
                          ;       Byte IDFBUF+3   Sektor Groesse  0=128 Bytes
                          ;                                       1=256 Bytes
                          ;                                       2=512 Bytes
                          ;                                       3=1024 Bytes
                          ;
                          ;       Wenn <A> aus dieser Routine mit NZ zurueckkommt, konnte
                          ;       das ID-Feld nicht gelesen werden. Dies kann 4 Gruende haben
                          ;       a) nicht formatierte Diskette   =Fehler bleibt konstant
                          ;       b) defekte Diskette             =wie A
                          ;       c) keine Seite 1(2) vorhanden   = Diskette nur 1-seitig
                          ;       d) falsche Dichte               = mit sd versuchen
```

```
                                    ;
                                    ; 2.    Aus der Sektorgroesse (IOFBUF+3) koennen Rueckschluesse
                                    ;       auf das Format geschlossen werden.
                                    ;
                                    ;
                                    ; Diese Art der 'Auto-Erkennung' kann natuerlich nicht alle
                                    ; Moeglichkeiten abdecken zumindest jedoch einige Standad-Formate
                                    ; erkennen. Is ein Format erkannt, muss der XDPH des entsprechenden
                                    ; Laufwerkes angepasst werden. Eine Anpassung ist notwendig fuer
                                    ;
                                    ;       a) die entsprechende DPB-Adresse
                                    ;       b) fuer die entsprechende XLT-Adresse
                                    ;
                                    ;
                                    ; In dieser Konfiguration lassen wir keine AUTO-Erkennung zu
                                    ; daher ....
                                    ;

0060' C9                            ret

006D'                       fd$read:

                                    ;
                                    ; lesen eines Sektors
                                    ; Aufruf mit folgenden Argumenten bereits gesetzt:
                                    ; Laufwerksnummer in (@drv)y
                                    ; Transferadresse in (@dma)
                                    ; Transferbank    in (@dbnk)
                                    ; Tracknummer     in (@trk)
                                    ; Sektornummer    in (@sect)
                                    ; Zeiger auf XDPH in   <DE>
                                    ;

006D' 21 01CF'                      ld      hl,read$msg     ; Operationsart
0070' 01 013C'                      ld      bc,read$sector
0073' 18 06                         jr      rw$common

0075'                       fd$write:

                                    ;
                                    ; schreiben eines Sektors
                                    ; Siehe hierzu FD$READ
                                    ;

0075' 21 01DB'                      ld      hl,write$msg    ; Operationsart
0078' 01 0144'                      ld      bc,write$sector

007B'                       rw$common:

                                    ;
                                    ; gemeinsamer Schreib/ Lese-Programmteil
                                    ;

007B' 22 0275'                      ld      (opname),hl     ; Art der Operation fuer Fehlermeldung
007E' C5                            push    bc              ; rette Funktion
007F' CD 014C'                      call    setup           ; berechne phys Track setze sso
0082' E1                            pop     hl              ; Adresse des UP's (READ-WRITE)
```

```
0083"                 more$retries:

0083" 06 0A                      ld      b,10            ; 10 Versuche zulassen

0085"                 retry$loop:

0085" C5                         push    bc              ; Schleifenzaehler
0086" E5                         push    hl              ; UP
0087" 3A 0270"                   ld      a,(cursel)      ; neuer select (ohne sso)
008A" 21 027A"                   ld      hl,oldsel       ; Zeiger auf 'letzen' select
008D" BE                         cp      (hl)
008E" 77                         ld      (hl),a          ; fuer's naechste mal ...
008F" 20 0A                      jr      nz,new$track    ; dann SEEK
0091" 3A 0000#                   ld      a,(@trk)        ; alter Track (logisch)
0094" 21 0279"                   ld      hl,oldtrk       ; Zeiger auf 'letzen' Track (logisch)
0097" BE                         cp      (hl)
0098" 77                         ld      (hl),a
0099" 28 09                      jr      z,same$track

009B"                 new$track:

009B" CD 00F4"                   call    check$seek      ; Track einstellen
009E" 01 0064                    ld      bc,100          ; 100ms
00A1" CD 01A6"                   call    delay           ; warten


00A4"                 same$track:

00A4" 3A 0272"                   ld      a,(curtrk)      ; physikalischer Track
00A7" D3 C1                      out     (fdctrk),a      ; setzen
00A9" 3A 0000#                   ld      a,(@sect)       ; physikalischen Sektor
00AC" D3 C2                      out     (fdcsec),a      ; setzen
00AE" E1                         pop     hl              ; UP
00AF" E5                         push    hl              ; ... retten fuer weiter Versuche
00B0" CD 0000#                   call    ipchl           ; ausfuehren
00B3" E1                         pop     hl              ;
00B4" C1                         pop     bc              ; Schleifenzaehler
00B5" C8                         ret     z               ; Fehlerfrei ...
00B6" C5                         push    bc
00B7" E5                         push    hl
00B8" CB 67                      bit     4,a             ; evtl RNF-Fehler
00BA" C4 00F4"                   call    nz,check$seek   ; dann nochmals SEEK
00BD" E1                         pop     hl              ; UP
00BE" C1                         pop     bc              ; Schleifenzaehler
00BF" 10 C4                      djnz    retry$loop      ; ... auch nach nochmaligem SEEK

                                 ;
                                 ; hierher wenn Fehler ...
                                 ;

00C1" 3A 0000#                   ld      a,(@ermde)      ; Pruefe ob Fehlermeldungen
00C4" 3C                         inc     a               ; FF wird 0
00C5" 28 2A                      jr      z,hard$error    ; ... wenn JA

                                 ;
                                 ; Fehlermeldung
                                 ;

```

```
00C7" E5                    push    hl              ; UP
00C8" CD 0000#              call    ?pderr          ; Fehlermeldung ausgeben
00CB" 2A 0275"              ld      hl,(opname)     ; lesen oder schreiben?
00CE" CD 0000#              call    ?pmsg           ; ausgeben
00D1" 3A 0273"              ld      a,(dskstat)     ; Fehlerstatus ruecklesen
00D4" 21 01C3"              ld      hl,error$table  ; Fehlertabelle

00D7" 5E            errml:  ld      e,(hl)          ; Fehlermeldung aus Tabelle
00D8" 23                    inc     hl
00D9" 56                    ld      d,(hl)          ; einlesen
00DA" 23                    inc     hl
00DB" 87                    add     a,a             ; Fehlerbits 1* links schieben
00DC" F5                    push    af              ;   und Zeichen + Flag retten
00DD" EB                    ex      de,hl           ; HL zeigt auf Fehlermeldung
00DE" DC 0000#              call    c,?pmsg         ;   diese Ausgeben
00E1" EB                    ex      de,hl           ; Zeiger zurueck
00E2" F1                    pop     af              ; und Fehlerbits
00E3" 20 F2                 jr      nz,errml        ; ... wenn noch weitere Fehler
00E5" 21 025D"              ld      hl,error$msg    ; nochmal?
00E8" CD 0000#              call    ?pmsg
00EB" CD 01B2"              call    conecho
00EE" E1                    pop     hl              ; UP
00EF" 28 92                 jr      z,more$retries  ;  nochmal ..

00F1"               hard$error:

00F1" 3E 01                 ld      a,1
00F3" C9                    ret


                            ;
                            ; ***************
                            ; Hilfsprogramme
                            ; ***************
                            ;

00F4"               check$seek:

                            ;
                            ; hier wird mit READ$ID geprueft ob der anzusprechende Track
                            ; dem eingestellten Track enstpricht
                            ; Sollte dies nicht der Fall sein wird die Kopfstellung
                            ; entsprechend korrigiert

00F4" CD 0130"              call    read$id         ; versuche ID-Feld zu lesen
00F7" 28 05                 jr      z,id$ok         ; ... wenn ok
00F9" CD 010D"              call    restore         ; sonst HOME
00FC" 06 00                 ld      b,0
00FE" 78            id$ok:  ld      a,b
00FF" D3 C1                 out     (fdctrk),a      ; Ins Trackregister
0101" 21 0272"              ld      hl,curtrk       ; Zeiger auf physikalischen Track
0104" BE                    cp      (hl)            ; beide gleich ?
0105" C8                    ret     z               ; dann fertig
0106" 7E                    ld      a,(hl)          ; sonst SOLL-Register
0107" D3 C3                 out     (fdcdat),a      ; ins Datenregister
0109" 06 10                 ld      b,seek          ; + SEEK Befehl
010B" 18 02                 jr      busy$cmd        ; ausfuehren

010D"               restore:
```

```
                                 ;
                                 ; hier wird der Kopf auf Track 0 eingestellt
                                 ;

    010D" 06 00                  ld      b,home          ; Laufwerk auf TRACK 0

    010F"               busy$cmd:

                                 ;
                                 ; fuer Laufwerke die schnell genug sind wirkliche 3ms zu steppen
                                 ; kann hier das MINI-BIT zurueckgesetzt werden - damit wird mit
                                 ; doppelter Geschwindigkeit gesteppt
                                 ; ACHTUNG: HOME und SEEK muessen dazu OHNE VERIFY-Bit sein
                                 ;

    010F" 3A 027B"               ld      a,(fflag)
    0112" F5                     push    af
    0113" CB AF                  res     5,a
    0115" D3 C4                  out     (fdcsel),a

    0117" 3A 0271"               ld      a,(curstep)     ; stepping-rate
    011A" B0                     or      b               ; einfuegen
    011B" D3 C0                  out     (fdccmd),a      ; Befehl ausgeben
    011D" DB C0         busy:    in      a,(fdccmd)
    011F" 0F                     rrca
    0120" 30 FB                  jr      nc,busy
    0122" DB C0         busyl:   in      a,(fdccmd)      ; nun abwarten
    0124" CB 47                  bit     0,a             ; bis NICHT mehr busy
    0126" 20 FA                  jr      nz,busyl
    0128" 47                     ld      b,a             ; rette Status
    0129" F1                     pop     af              ; Select
    012A" D3 C4                  out     (fdcsel),a
    012C" 78                     ld      a,b             ; Status
    012D" E6 90                  and     10010000b       ; Maskiere READY & RNF-SEEK
    012F" C9                     ret

    0130"               read$idf:

                                 ;
                                 ; lesen des IF-Feldes in den IDF-Puffer
                                 ;

    0130" 21 006D'               ld      hl,idfbuf       ; Braucht eigenen Puffer
    0133" 06 C4                  ld      b,readid
    0135" E5                     push    hl
    0136" CD 0011'               call    getdata
    0139" E1                     pop     hl
    013A" 46                     ld      b,(hl)          ; Tracknummer
    013B" C9                     ret

    013C"               read$sector:

                                 ;
                                 ; lesen eines physikalischen Sektors
                                 ; Erwartet Disk-Track-Sektor-DMA gesetzt
                                 ;

```

```
013C" 2A 0000#            ld      hl,(@dma)       ; DMA
013F" 06 88               ld      b,reads
0141" C3 0011'            jp      getdata
0144"           write$sector:
                          ;
                          ; schreiben eines physikalischen Sektors
                          ;
0144" 2A 0000#            ld      hl,(@dma)
0147" 06 A8               ld      b,writes
0149" C3 0036'            jp      putdata
014C"           setup:
                          ;
                          ; hole SELECT und STEPPINGRATE aus XDPH
                          ; LEGE XDPH-Adresse ab
                          ;
014C" EB                  ex      de,hl
014D" 22 026E"            ld      (curdph),hl     ; Ablegen
0150" 2B        setup0:   dec     hl              ; Select
0151" 5E                  ld      e,(hl)          ; lesen
0152" 2B                  dec     hl              ; Steppingrate
0153" 56                  ld      d,(hl)
0154" ED 53 0270"         ld      (cursel),de     ; ablegen
                          ;
                          ; physikalischen Track aus logischen Track berechen
                          ; und damit side-select
                          ;
0158" 3A 0000#            ld      a,(@trk)        ; zuvor logischen TRACK einlesen
015B" CB 7B               bit     7,e             ; einseitiges Laufwerk?
015D" 20 06               jr      nz,setx         ; wenn JA
015F" CB 6B               bit     5,e             ; evtl 8" sd?
0161" 28 02               jr      z,setx          ; wenn JA
0163" B7                  or      a               ; reset CARRY
0164" 1F                  rra                     ; /2
0165" 32 0272"  setx:     ld      (curtrk),a      ; ist physikalischer Track
0168" 3E 00               ld      a,0             ; sso-Flag schuetzen ...
016A" 30 04               jr      nc,setup1       ; Seite 0(1)
016C" CB CF               set     1,a             ; sso-bit setzen
016E" CB FB               set     7,e             ; auch in SELCOD side-select einbringen
0170" 32 006C'  setup1:   ld      (ssoflag),a     ; sso-Flag ablegen
0173" 7B                  ld      a,e             ; select
0174" D3 C4               out     (fdcsel),a      ; ausgeben
0176" 32 027B"            ld      (fflag),a       ; rette wirklichen Selekt
                          ;
                          ; Testen ob Diskette eingelegt ist
                          ;
0179" 21 0000             ld      hl,0            ; time_out
017C" 2B        setup2:   dec     hl              ; time_out -1
017D" 7C                  ld      a,h
```

```
017E" B5                    or      l
017F" 28 06                 jr      z,setup3
0181" DB C0                 in      a,(fdccmd)
0183" 07                    rlca                    ; READY?

                            ;
                            ; Anmerkung: bei festverdrahtetem READY ist diese Routine
                            ; sinnlos !!!! dann evtl Index-Loch abfragen ... nach 250ms Delay
                            ;

0184" 38 F6                 jr      c,setup2
0186" C9                    ret

0187" 3A 0000#      setup3: ld      a,(@drv)        ; Laufwerk
018A" C6 41                 add     a,'A'           ; .. in ASCII
018C" 32 01F4"              ld      (errdsk),a      ; in
018F" 21 01EB"              ld      hl,nodisk       ; Fehlermeldung
0192" CD 0000#              call    ?pmsg
0195" 21 025D"              ld      hl,error$msg    ; ... nochmal
0198" CD 0000#              call    ?pmsg
019B" CD 01B2"              call    conecho
019E" C2 0000               jp      nz,0            ; ... nein  Warmstart
01A1" 2A 026E"              ld      hl,(curdph)     ; XDPH zuruecklesen
01A4" 18 AA                 jr      setup0          ; und nochmal das Ganze

01A6"               delay:

                            ;
                            ; Warteschleife mit Zaehler in <BC>
                            ; Je Einheit wird 1 ms gewartet
                            ;

01A6" 0B                    dec     bc
01A7" C5                    push    bc
01A8" 06 B9                 ld      b,185           ; bei 4 MHz
01AA" 10 FE                 djnz    $
01AC" C1                    pop     bc
01AD" 78                    ld      a,b
01AE" B1                    or      c
01AF" 20 F5                 jr      nz,delay
01B1" C9                    ret

01B2"               conecho:

                            ;
                            ; Verlangt Zeicheneingabe von Konsole
                            ; Zeichen wird zum Grossbuchstaben konvertiert
                            ; und das Z-Flag gesetzt wenn J oder Y gefunden wurde
                            ;

01B2" CD 0000#              call    ?conin          ; Zeichen abfragen
01B5" F5                    push    af              ; retten
01B6" 4F                    ld      c,a
01B7" CD 0000#              call    ?cono           ; ECHO
01BA" F1                    pop     af
01BB" E6 5F                 and     5fh             ; mache Grossbuchstaben daraus
01BD" FE 4A                 cp      'J'
01BF" C8                    ret     z
```

```
01C0" FE 59                cp      'Y'
01C2" C9                   ret
                           cseg                    ; im gemeinsamen Bereich
                           ;
                           ; Anmerkung: JP benoetigt weniger Zeit als JR -
                           ; und ohne DMA ist diese Routine recht Zeitkritisch
                           ;
0011' 3A 0000#    getdata:ld      a,(@dbnk)        ; Zielbank
0014' CD 0000#            call    ?bank            ; setzen
0017' 3A 006C'            ld      a,(ssoflag)      ; sso-Bit
001A' B0                  or      b                ; in Befehl einfuegen
001B' 0E C3               ld      c,fdcdat         ; Datenport
001D' D3 C0               out     (fdccmd),a       ; Befehl ausgeben
001F' DB C4       getd1:  in      a,(fdcsel)       ; abwarten bis
0021' 07                  rlca                     ; BUSY aktiv
0022' 30 FB               jr      nc,getd1
0024' DB C0       getd2:  in      a,(fdccmd)       ; Status abfragen
0026' CB 4F               bit     1,a              ; DRQ?
0028' 28 05               jr      z,getd3          ; wenn noch nicht ...
002A' ED A2               ini                      ;   sonst Daten einlesen
002C' C3 0024'            jp      getd2            ; und weiter abfragen
002F' CB 47       getd3:  bit     0,a              ; nicht mehr BUSY?
0031' C2 0024'            jp      nz,getd2
0034' 18 23               jr      rw$done
0036' 3A 0000#    putdata:ld      a,(@dbnk)
0039' CD 0000#            call    ?bank
003C' 3A 006C'            ld      a,(ssoflag)
003F' B0                  or      b
0040' 0E C3               ld      c,fdcdat
0042' D3 C0               out     (fdccmd),a
0044' DB C4       putd1:  in      a,(fdcsel)
0046' 07                  rlca
0047' 30 FB               jr      nc,putd1
0049' DB C0       putd2:  in      a,(fdccmd)
004B' CB 4F               bit     1,a
004D' 28 05               jr      z,putd3
004F' ED A3               outi
0051' C3 0049'            jp      putd2
0054' CB 47       putd3:  bit     0,a
0056' C2 0049'            jp      nz,putd2
0059' F5          rw$done:push    af               ; Rette Fehlercode
005A' 3A 0000#            ld      a,(@cbnk)
005D' CD 0000#            call    ?bank
0060' F1                  pop     af               ; Fehlercode
0061' 32 0273"            ld      (dskstat),a      ; Status ablegen
0064' E6 FC               and     11111100b        ; Fehler maskieren
0066' C9                  ret
0067'             motoff:
```

```
                             ;
                             ; Motorabschaltung
                             ;

0067' 3E 40                  ld      a,01000000b     ; Motor
0069' D3 C4                  out     (fdcsel),a      ; abschalten
006B' C9                     ret

                             dseg

                             ;
                             ; ****************
                             ; Tabellen
                             ; ****************
                             ;

01C3"                  error$table:

01C3" 01F6"                  dw      b7$msg
01C5" 0204"                  dw      b6$msg
01C7" 0213"                  dw      b5$msg
01C9" 0225"                  dw      b4$msg
01CB" 023B"                  dw      b3$msg
01CD" 024F"                  dw      b2$msg

                             ;
                             ; ****************
                             ; Text Bereich
                             ; ****************
                             ;

01CF" 2C 20 62 65 read$msg:db     ', beim Lese','n'+80H
01D8" 2C 20 62 65 write$msg:db    ', beim Schreibe','n'+80H
01EB" 4C 61 75 66 nodisk: db      'Laufwerk '
01F4" 41 3A       errdsk: db      'A:'
01F6" 20 6E 69 63 b7$msg: db      ' nicht bereit',' '+80H
0204" 20 53 63 68 b6$msg: db      ' Schreibschutz',' '+80H
0213" 20 4B 6F 6E b5$msg: db      ' Kontrollerfehler',' '+80H
0225" 20 44 61 74 b4$msg: db      ' Datei nicht gefunden',' '+80H
023B" 20 66 61 6C b3$msg: db      ' falsche Pruefsumme',' '+80H
024F" 20 44 61 74 b2$msg: db      ' Datenverlust',' '+80H

025D"                  error$msg:

025D" 2D 20 6E 6F            db      '- nochmal? (J/N)',' '+80H

                             ;
                             ; ****************
                             ; Variablenbereich
                             ; ****************
                             ;

026E"   0002      curdph: ds      2               ; momentane XDPH-Adresse
0270"   0001      cursel: ds      1               ; momentaner Select
0271"   0001      curstep:ds      1               ; momentane stepping-rate
0272"   0001      curtrk: ds      1               ; physikalischer Sektor
0273"   0002      dskstat:ds      2               ; Fehlercode
0275"   0002      opname: ds      2               ; Operationsart
0277"   0002      lastop: ds      2               ; LOW=DRIVE HIGH=1 wenn read 2=wenn write
```

```
0279"   0001      oldtrk: ds      1
027A" FF          oldsel: db      -1              ; ungueltig beim Start
027B" 00          fflag:  db      0

                          cseg

006C'   0001      ssoflag:ds      1
006D'   0006      idfbuf: ds      6

                          end
0 Error(s) Detected. 115 Program Bytes. 636 Data Bytes.
226 Symbols Detected.
```

Cross Reference:

```
0028  ??0005            153   153   153
0004  ??0006            153   153   153   154
000F  ??0007            153   154
0000  ??0008            153   153   153   153   153   155
0185  ??0009            153   153   156
00FF  ??000A            153   157
00F0  ??000B            153   158
0000  ??000C            153   158
0040  ??000D            153   159
0003  ??000E            153   153   153   153   153   153   153   161
0007  ??000F            153   161
0004  ??0010            153   153   153   153   153   153   153   153   153
                         153   153   153   153   153   153   153   153   153
                         153   153   153   153   153
F000  ?ALL              153   153   153   153   153   153   153   153   153
                         153   153
005E# ?BANK              19   590   609   628
01B3# ?CONIN             34   571
01B8# ?CONO              34   574
0000# ?CONST             35
0000  ?GCDM             181   181   181
0001  ?GCDN             181   181   181   181
0000  ?GCDR             181   181   181
0000  ?GCDX             181   181
0005  ?NELTS            181   182   182   182   183   183   183   184   184
                         184   185   185   185   186   186   186   186
0005  ?NELTST           181   181   186
0001  ?NXTBAS           181   186   186   186
0001  ?NXTSEC           181   181   182   182   182   182   183   183   183
                         183   184   184   184   184   185   195   185   195
                         186   186   186   186
0000# ?PDEC              34
00C9# ?PDERR             34   348
0199# ?PMSG              34   350   361   366   538   540
1860  ?SIZE             153   153
0000# ?WBOOT             34
005E# @CBNK              21   627
0037# @DBNK              21   589   608
0145# @DMA               20   463   473
0188# @DRV               19   534
0000" @DTBL              41    55
00C2# @ERMDE             28   339
00AA# @SECT              20   319
0159# @TRK               20   302   497
024F" B2$MSG            659   676
0238" B3$MSG            658   675
0225" B4$MSG            657   674
0213" B5$MSG            656   673
0204" B6$MSG            655   672
01F6" B7$MSG            654   671
0110" BUSY              429   431
010F" BUSY$CMD          402   412
0122" BUSY1             432   434
00F4" CHECK$SEEK        310   330   382
01B2" CONECHO           367   541   563
026E" CURDPH            485   543   588
0270" CURSEL            297   490   689
```

```
Cross Reference:

0271" CURSTEP          426   690 .
0272" CURTRK           317   396   504   691
01A6" DELAY            312   545   560
0000  DPB              152
0000' DPB80D            90   110   152
0000  DPH               86   106
0273" DSKSTAT          351   630   692
01F4" ERRDSK           536   670
00D7" ERRM1            354   364
025D" ERROR$MSG        365   539   678
01C3" ERROR$TABLE      352   652
006B" FD$INIT           83   103   193
006C" FD$LOGIN          82   102   204
006D" FD$READ           81   101   251
0075" FD$WRITE          80   100   268
002A" FDA               55    86
004D" FDB               56   106
0000  FDC               57   120
00C0  FDCCMD           428   429   432   523   594   598   613   617
00C3  FDCDAT           400   593   612
00C2  FDCSEC           320
00C4  FDCSEL           424   437   511   595   614   641
00C1  FDCTRK           318   395
0000  FDD               58   121
0000  FDE               59   122
0000  FDF               60   123
0000  FDG               61   124
0000  FDH               62   125
0000  FDI               63   126
0000  FDJ               64   127
0000  FDK               65   128
0000  FDL               66   129
0000  FDM               67   130
0000  FDN               68   131
0000  FDO               69   132
0000  FDP               70   133
027B" FFLAG            421   512   697
0000  GCD              181
001F' GETD1            595   597
0024' GETD2            598   602   605
002F' GETD3            600   604
0011' GETDATA          451   465   589
00F1" HARD$ERROR       341   371
0000  HOME             410
00FE" ID$OK            391   394
006D' IDFBUF           448   702
00B1# IPCHL             22   323
0277" LASTOP           694
0020  MINIS             85   105
0083" MORE$RETRIES     289   369
0067' MOTOFF            41   634
0000  NEIN             120   121   122   123   124   125   126   127   128
                        129   130   131   132   133
009B" NEW$TRACK        301   308
01EB" NODISK           537   669
027A" OLDSEL           298   696
0279" OLDTRK           303   695
0275" OPNAME           284   349   693
```

Cross Reference:

```
0044' PUTD1              614  616
0049' PUTD2              617  621  624
0054' PUTD3              619  623
0036' PUTDATA            475  608
0130" READ$IDF           390  442
01CF" READ$MSG           264  667
013C" READ$SECTOR        265  456
00C4  READID             449
0088  READS              464
010D" RESTORE            392  404
0085" RETRY$LOOP         293  333
007B" RW$COMMON          266  278
0059' RW$DONE            606  626
00A4" SAME$TRACK         306  315
0010  SEEK               401
0001  SELA                85
0002  SELB               105
014C" SETUP              286  477
0150" SETUP0             486  544
0170" SETUP1             506  509
017C" SETUP2             519  531
0187" SETUP3             522  534
0165" SETX               499  501  504
0000  SKEW               180
006C' SSOFLAG            509  591  610  701
01DB" WRITE$MSG          275  668
0144" WRITE$SECTOR       276  467
00A8  WRITES             474
0066" XLT5                87  107  180
```

Kapitel 10 Das BIOS-Segment BOOT.

In diesem BIOS-Programm-Modul werden 5 Aufgaben erledigt:

1. Die Zuweisung der physikalischen Schnittstellen an die logischen Schnittstellen, zumindest soweit dies CONIN- und CONOUT betrifft, üblicherweise aber auch AUXIN, AUXOUT und LST.

2. Alle Kaltstart-Initialisierungen. Dazu gehören die (evtl. noch nicht vorgenommene) Hardware-Initialisierungen, die Schnittstellen-Initialisierungen in CHARIO und DISKIO; die Initialisierung einer evtl. vorhandene Uhr und die Ausgabe der Systemmeldung.

3. In diesem Programm-Modul befindet sich der Treiber um den Programmteil CCP.COM von Diskette zu lesen und in die TPA zu transferieren sowie in einen passenden (reservierten) Bankbereich.

4. Zum Warmstart stellt das Modul ein Unterprogramm zur Verfügung, welches das Programm CCP.COM aus einer Bank (siehe Punkt 3) in die TPA transferiert.

5. Alle notwendigen Uhrentreiber (falls vorhanden) werden zur Verfügung gestellt.

Die Einzelheiten können dem nachfolgenden Listing entnommen werden:

```
                                  MACLIB  DEFAULT.INC      ;Vereinbarungen
                          ;*****************************************************************
                          ;*                                                               *
                          ;* rel 1.0 vom 07.01.85                                          *
                          ;*                                                               *
                          ;* Die hauptsaechliche Aufgabe des BOOT-Moduies ist die          *
                          ;* zusaetzliche hardware-Initialisierung und das Laden           *
                          ;* des CCP - im gebankten System das Umladen von Bank zu Bank    *
                          ;* In diesem Programmsegment ist auch die UHR angesiedelt        *
                          ;*                                                               *
                          ;* geschrieben von RAOUL O. KOERBER                              *
                          ;* nach Unterlagen von DRI                                       *
                          ;*                                                               *
                          ;*****************************************************************

                                  ;
                                  ; ****************
                                  ; Vereinbarungen
                                  ; ****************
                                  ;

            0005          bdos    equ     0005H

                                  ;
                                  ; ****************
                                  ; Systemadressen
                                  ; ****************
                                  ;

                                  public  ?init,?ldccp,?rlccp,?time
                                  extrn   ?pmsg,?conin
                                  extrn   @civec,@covec,@aivec,@aovec,@lovec
                                  extrn   @cbnk,?bnksl
                                  extrn   ?xmove,?move            ; fuer ?rlccp und ?ldccp ...
                                  extrn   tbaud                   ; Baudrate Terminal
                                  public  inisub

                                  dseg                            ; BANK 0 !

                                  ;
                                  ; ********************************
                                  ; Programmstart
                                  ; ********************************
                                  ;


                                  ;
                                  ; ****************
                                  ; Hardwareinit
                                  ; ****************
                                  ;

 0000"                    ?init:


                                  ;
                                  ; ?init wird beim Kaltstart (BOOT) aufgerufen jedoch nicht
                                  ; beim Warmstart.
                                  ; Hier koennen spezielle Initialisierungen vorgenommen
                                  ; werden die nicht vorher vorgenommen wurden
```

```
                                     ;
                                     ; **** Zuweisung der Einheiten
                                     ; diese Zuweisung ist Benutzerabhaengig und
                                     ; muss evtl angepasst werden.
                                     ; Aenderungen koennen ON-LINE mit Device vor-
                                     ; genommem werden ....
                                     ; Siehe hierzu CHARIO.MAC
                                     ;

0000" 21 8000                        ld      hl,1000000000000000b    ; KEY als EINGABE
0003" 22 0000#                       ld      (@civec),hl             ; Vektor ablegen
0006" 26 40                          ld      h,01000000b             ; GDP als AUSGABE
0008" 22 0000#                       ld      (@covec),hl             ; Vektor ablegen
000B" 26 10                          ld      h,00010000b             ; AUX auf SER-Karte
000D" 22 0000#                       ld      (@aivec),hl             ; als AUX  - Eingabe
0010" 22 0000#                       ld      (@aovec),hl             ;        und Ausgabe
0013" 26 04                          ld      h,00000100b             ; als PRINT CENTRONICs
0015" 22 0000#                       ld      (@lovec),hl             ;

                                     ;
                                     ; Zuweisung soweit erledigt - Erweiterung durch setzen
                                     ; der entsprechenden BITs  oder durch DEVICE.COM
                                     ;

                                     ;
                                     ; **** Standard initialisierung
                                     ;

0018" 21 005A"                       ld      hl,inittable
001B" CD 0000'                       call    inisub

                                     ;
                                     ; sonst noch was ? - dann hier einfuegen ...
                                     ;

001E" 21 0024"                       ld      hl,signon$msg           ; Begruessung
0021" C3 0000#                       jp      ?pmsg

                                     cseg                            ; gemeinsamer Bereich

0000'                        inisub:


                                     ;
                                     ; Hilfsprogramm zur initialisierung
                                     ; Bei Aufruf zeigt <HL> auf eine Zeichenkette
                                     ; 1. Zeichen = Anzahl der Kombinationen PORT/Zeichen
                                     ; danach PORT-Adresse gefolgt von INIT-Zeichen
                                     ;

0000' 46                             ld      b,(hl)                  ; Laenge des INITstrings(/2)
0001' 78                             ld      a,b                     ; Test ob NULL-String
0002' B7                             or      a                       ; wenn NULL ...
0003' C8                             ret     z                       ; dann fertig
0004' 23                     inisl:  inc     hl                      ; naechster Wert
0005' 4E                             ld      c,(hl)                  ; Initport
0006' 23                             inc     hl                      ; naechster Wert
0007' 7E                             ld      a,(hl)                  ; Initwert
0008' ED 79                          out     (c),a                   ; Wert senden
```

```
000A' 10 F8             djnz    inisl
000C' C9                ret

                ;
                ; ******************************
                ; CCP kalt- und Warmstartlader
                ; ******************************
                ;

                ;
                ; ***************
                ; Kaltstart
                ; ***************
                ;

000D'           ?ldccp:

                ;
                ; lade CCP (erstmals von Diskette) in TPA
                ; Prinzipell kann CCP.COM auch hinter CPMLDR
                ; gehaengt werden und beim booten mit auf Bank 0 geladen
                ; werden ... von dort dann direkt in den Bereich D000..DFFF
                ; ( also direkt unter dem gemeinsamen RAM-Bereich gebankt
                ;
000D' AF                xor     a
000E' 32 008C'          ld      (ccp$fcb+15),a          ; Extent in FCB = 0
0011' 67                ld      h,a
0012' 6F                ld      l,a                     ; HL = 0
0013' 22 009D'          ld      (fcb$nr),hl             ; Start bei FILE-Beginn
0016' 11 007D'          ld      de,ccp$fcb              ; FCB
0019' CD 0061'          call    _open                   ; OPEN FILE via BDOS in CPMLDR
001C' 3C                inc     a                       ; Fehler ?
001D' 28 24             jr      z,no$ccp                ; ... kein File CCP.COM !
001F' 11 0100           ld      de,100h                 ; CCP-Start
0022' CD 0064'          call    _setdma
0025' 11 0080           ld      de,128                  ; max 16k bytes ...
0028' CD 0067'          call    _setmulti
002B' 11 007D'          ld      de,ccp$fcb
002E' CD 006A'          call    _read                   ; nun einlesen

                ;
                ; wenn gebankt wird CCP gleichzeitig nach Bank 0 geladen
                ; damit Warmstart ohne Diskettenzugriff moeglich ist ...
                ;

0031' 01 0001           ld      bc,ccpbnk shl 8 + 1     ; B=Ziel - C=Quelle
0034' CD 0000#          call    ?xmove                  ; ... transfer mit xmove-Power
0037' 21 C000           ld      hl,ccpis                ; Ziel in ccpbnk
003A' 11 0100           ld      de,tpa                  ; Quelle
003D' 01 0D00           ld      bc,ccplen               ; muesste jetzt reichen ...
0040' C3 0000#          jp      ?move

                ;
                ; ***************
                ; Fehler - Kein CCP
                ; ***************
                ;
```

```

0043'               no$ccp:


                                  ;
                                  ; der File CCP.COM ist nicht auf der Diskette ( zu finden )
                                  ; das kann zwei Ursachen haben - er ist wirklich nicht da
                                  ; oder mit den CP/M-Treibern passt was nicht ....
                                  ;

0043' 21 006F'                    ld      hl,ccp$msg
0046' CD 0000#                    call    ?pmsg                ; Fehlermeldung an Konsole
0049' CD 0000#                    call    ?conin               ; als Warteschleife ...

                                  ;
                                  ; was nuetzt eine endlose Schleife ...
                                  ;

004C' 18 BF                       jr      ?ldccp               ; den ewigen Sprung

                                  ;
                                  ; ***************
                                  ; Warmstart (Reload)
                                  ; ***************
                                  ;

004E'               ?rlccp:

                                  ;
                                  ; nachladen des CCP aus Bank oder von Diskette
                                  ;

004E' 01 0100                     ld      bc,1 shl 8 + ccpbnk  ; B=ZIEL - C=Quelle
0051' CD 0000#                    call    ?xmove               ; mit XMOVE-Power
0054' 21 0100                     ld      hl,tpa               ; Zieladresse in Bank 1
0057' 11 C000                     ld      de,ccpis             ; Quelladresse in ccpbnk
005A' 01 0D00                     ld      bc,ccplen            ; Laenge
005D' C3 0000#                    jp      ?move

                                  ;
                                  ; ******************************
                                  ; UHR
                                  ; ******************************
                                  ;

0060'               ?time:

                                  ;
                                  ; Hard- oder Softwareuhr - z.Z. nicht implementiert
                                  ; Wenn Implementiert entsprechende LABELS in SCB.ASM
                                  ; zur Zeitablege ansprechen ... Anpassung uebernimmt
                                  ; GENCPM ...
                                  ;

0060' C9                          ret

                                  ;
                                  ; ******************************
                                  ; BDOS-Interface
```

```
                                ; ********************************
                                ;
                                ;
                                ; CP/M BDOS Interface-Funktionen
                                ;

0061'                   _open:

0061' 0E 0F                     ld      c,15
0063' 21                        db      21h                     ; SKIP

0064'                   _setdma:

0064' 0E 1A                     ld      c,26
0066' 21                        db      21h                     ; SKIP

0067'                   _setmulti:

0067' 0E 2C                     ld      c,44
0069' 21                        db      21h                     ; SKIP

006A'                   _read:

006A' 0E 14                     ld      c,20
006C' C3 0005                   jp      bdos


                                ;
                                ; ********************************
                                ; Textbereich - HALLO ....
                                ; ********************************
                                ;

                                dseg                            ; Bank 0

0024"                   signon$msg:

0024" 1A 0A                     db      cls,lf
0026" 2A 2A 2A 20               db      '*** CP/M VERS 3.1',cr,lf
0039" 20 20 20 20               db      '    mit NDR-Computer-BIOS ',rel,'.',rev,usr

                                ;
                                ; hier kann Benutzer-Message eingefuegt werden
                                ;

0057" 0D 0A 8A                  db      cr,lf,lf+80H

                                cseg                            ; gemeinsamer Bereich

006F'                   ccp$msg:

006F' 0D 0A 6B 65               db      cr,lf,'kein CCP.CO','M'+80H

                                ;
                                ; ****************
                                ; CCP-FBC
                                ; ****************
                                ;

```

```
007D'             ccp$fcb:

007D' 01 43 43 50          db      1,'CCP     COM',0,0,0,0
008D'   0010               ds      16
0090' 00 00 00    fcb$nr:  db      0,0,0

                           dseg                            ; bank 0

                           ;
                           ; ********************************
                           ; Initialisierungstabelle
                           ; ********************************
                           ;

005A"             inittable:

                           ;
                           ; Eintragung der INIT-Werte in der Folge PORT - WERT
                           ; hier kann auch das ( oder mehrere ) Parallel-Drucker
                           ; Interface (nach) initialisiert werden ...
                           ;

005A" 00                   db      i$len/2          ; Laenge

005B"             i$tabl:                           ; im Grundprogramm liegt nichts vor ..
        0000      i$len    equ     $-i$tabl

                           cseg

                           ;
                           ; ****************
                           ; Fehlerkorrektur
                           ; ****************
                           ;
                           ;
                           ; GENCPM kann unter bestimmten Umstaenden einen
                           ; Fehler bei der Berechnung der Startadresse machen
                           ; dies wird hier korrigiert.
                           ; BOOT MUSS JEDOCH UNBEDINGT DIE LETZTE DATEI IN DER
                           ; LINK-KETTE SEIN !!!
                           ;

00A0'   0100               ds      100h

                           end
0 Error(s) Detected. 416 Program Bytes. 91 Data Bytes.
132 Symbols Detected.
```

Cross Reference:

```
0000# ?BNKSL          33
004A# ?CONIN          31   186
0000" ?INIT           30    52
000D' ?LDCCP          30   132   192
005E# ?MOVE           34   168   211
0047# ?PMSG           31    95   185
004E' ?RLCCP          30   200
0060' ?TIME           30   219
0052# ?XMOVE          34   164   207
000E# @AIVEC          32    73
0011# @AOVEC          32    74
0000# @CBNK           33
0004# @CIVEC          32    69
0009# @COVEC          32    71
0016# @LOVEC          32    76
0005  BDOS            22   257
007D' CCP$FCB        143   147   155   291
006F' CCP$MSG        184   281
0000  CCPBNK         163   206
C000  CCPIS          165   209
0D00  CCPLEN         167   210
001A  CLS            269
000D  CR             271   277   283
009D' FCB$NR         146   295
0000  I$LEN          313   316
005B" I$TAB1         315   316
0004' INIS1          112   117
0000' INISUB          36    88    99
005A" INITTABLE       87   305
000A  LF             269   271   277   277   283
0043' NO$CCP         150   176
0031  REL            272
0030  REV            272
0024" SIGNON$MSG      94   267
0000# TBAUD           35
0100  TPA            166   208
0030  USR            272
0061' _OPEN          148   239
006A' _READ          156   254
0064' _SETDMA        152   244
0067' _SETMULTI      154   249
```

Kapitel 11 Zuweisungen - Vereinbarungen - Macros

In einem doch recht komplexen Programm wie dem BIOS für CP/M 3, ist es vorteihaft immer wiederkehrende Vereinbarungen und Label-Namen für Daten in einer getrennten Datei zusammenzufaßen, die vor dem Assemblerlauf als INClude Datei (Einfügung) aufgerufen werden kann. Im gegebenen Beispiel handelt es sich um zwei Dateien:

1. DEFAULT.INC sie enthält alle Systemdaten, Portadressen und Befehlscodes an spezielle Bausteine.

2. MODEBAUD.INC sie enthält Daten zur Baudrate und Schnittstellentypisierung in CHARIO

dazu kommt noch die MACRO-Datei:

3. CPM3.INC hier sind alle Macros zusammengefaßt, die von DISKIO zur Generierung der Programmteile DPH,DPB und SKEW benötigt werden.

   Die entsprechenden Daten könnten sicherlich auch per Hand eingegeben werden - Macros sind hierzu aber einfacher und betriebssicherer zu handhaben.

Die nachfolgenden Listings zeigen alle entsprechenden Details:

```
Z80ASM SuperFast Relocating Macro Assembler 20 Dec 85 09:45 Z80ASM 1.24 Page   1
DEFAULT Z80 20 Dec 85 09:41

                    ;******************************************************************
                    ;*                                                                *
                    ;*  NDR-Computer Z80-CPM3                                         *
                    ;*  rel 1.0 vom 07.01.85                                          *
                    ;*                                                                *
                    ;* Dieses Programmsegment enthaelt alle Vorgabewerte und          *
                    ;* Konstanten, die mehrmals in anderen Programmteilen auf-        *
                    ;* tauchen oder der Steuerung eines Assemblerdurchlaufes          *
                    ;* dienen. Im Normalfalle muss nur dieses Programmsegment         *
                    ;* geaendert werden um 'andere' CP/M+ Systeme zu erstellen        *
                    ;*                                                                *
                    ;* In diesem Programmsegment werden auch alle Aenderungen         *
                    ;* nachgehalten die im Laufe der Zeit an den einzelnen            *
                    ;* Unterprogrammen vorgenommen wurden ... wir empfehlen dem       *
                    ;* Anwender dies beizubehalten                                    *
                    ;*                                                                *
                    ;* Dieses Programmsegment ist NUR sinnvoll wenn neu assembliert   *
                    ;* werden soll - es dient sonst lediglich der allgemeinen         *
                    ;* Information                                                    *
                    ;*                                                                *
                    ;* geschrieben von RAOUL O KOERBER                                *
                    ;*                                                                *
                    ;******************************************************************

                            ; ********************************
                            ; Fehler und Aenderungsrapport
                            ; ********************************

         0031       rel     equ     '1'             ; Haupt'Ausgabe'
         0030       rev     equ     '0'             ; wichtige Revision
         0030       usr     equ     '0'             ; Benutzerrevision

                    ;******************************************************************
                    ;*                                                                *
                    ;* am           wo      was                                       *
                    ;* -------------------------------------------------------------- *
                    ;*                                                                *
                    ;******************************************************************

                            ; ********************************
                            ; Vorgabewerte
                            ; ********************************

                            ; ****************
                            ; Assembler
                            ; ****************

         FFFF       ja      equ     -1              ; Bedingungsflags
         0000       nein    equ     not ja
         FFFF       banked  equ     ja

                            ; ****************
                            ; ASCII
                            ; ****************

         0000       null    equ     0
         0011       ctlq    equ     'Q'-'@'
         0013       ctls    equ     'S'-'@'
```

```
0007    bell    equ     07h
0008    bs      equ     08h         ; Backspace
0009    ht      equ     09h         ; Cursor rechts ( Hor-Tab)
000A    lf      equ     0ah         ; Zeilenvorschub ( linefeed)
000B    vt      equ     0bh         ; Cursor 'rauf ( Ver-Tab)
000C    ff      equ     0ch         ; Cursor rechts
000D    cr      equ     0dh         ; Return
0016    syn     equ     16h         ; Cursor eine Zeile 'runter
001A    cls     equ     1ah         ; CLS Bildschrim loeschen
001B    esc     equ     1bh         ; ESC-Zeichen
001C    fs      equ     1ch         ; CUROFF
001D    gs      equ     1dh         ; CURON
001E    chome   equ     1eh         ; Cursor HOME
001F    newline equ     1fh         ; CRLF
0020    blank   equ     20h

                ; ****************
                ; Systemadressen
                ; ****************

0100    tpa     equ     00100h      ; Start des TPA-Bereiches
0100    ccp     equ     tpa         ; Einsprung in CCP
4000    boram   equ     04000h      ; Start RAM in BOOT-karte
F400    monseg  equ     0f400h      ; Start des Monitorsegmentes
FC00    ioseg   equ     0fc00h      ; IO-Segment USER-AREA

                ; nachfolgende UP's rufen Programmsegmente in MONIO
                ; (den IO-Segment von ELMON ) auf - welche ihrerseits
                ; auf das BOOT-ROM-Segment in FLOMON zurueckgreifen.
                ; Diese (etwas umstaendliche) Prozedur ist fuer CP/M3
                ; notwendig, da hier der IO-Einsprung aus verschiedenen
                ; Banken erfolgen kann - FLOMON dies aber nicht unterstuetzt
                ; gleichzeitig wurde der Vorteil einer 3k groesseren TPA
                ; erreicht

FC00    econist equ     0fc00h      ; Konsolenstatus (EIN) aus MONIO
FC03    econin  equ     0fc03h      ; Konsoleneingabe aus MONIO
FC06    econout equ     0fc06h      ; Standart-Ausgabe
FC24    esetbnk equ     0fc24h      ; Bankumschaltung (mit ROM-Ausschaltung)
FC27    efloppy equ     0fc27h      ; Floppy-Treiber
FC34    ecurbnk equ     0fc34h      ; Bankadresse (gueltige)
FC35    e@dbnk  equ     0fc35h      ; @DBNK in MONIO
FC36    e@cbnk  equ     0fc36h      ; @CBNK in MONIO

                ; ****************
                ; PORTS
                ; ****************

                ; **** Centronic

0040    ioec    equ     40h         ; Basisport
0048    iodat   equ     ioec+8      ; Datenport
0049    iocmd   equ     ioec+9      ; Statusport

                ; **** SER mit UART 6551 als Terminal

00FC    ubas    equ     0fch        ; Basisport
00FC    udat    equ     ubas        ; Datenport
```

```
00FD    usta    equ     udat+1
00FE    ucmd    equ     udat+2
00FF    ucon    equ     udat+3

                ; **** SER mit UART 6551 als SERieller Drucker

00EC    ubas1   equ     0ech            ; Basisport
00EC    udat1   equ     ubas1
00ED    usta1   equ     udat1+1
00EE    ucmd1   equ     udat1+2
00EF    ucon1   equ     udat1+3

                ; **** SER mit UART 6551 als AUX

00CC    ubas2   equ     0cch            ; Basisport
00CC    udat2   equ     ubas2
00CD    usta2   equ     ubas2+1
00CE    ucmd2   equ     ubas2+2
00CF    ucon2   equ     ubas2+3

                ; **** Serielle Schnittstelle KAS

00CA    casbas  equ     0cah
00CA    cmdcas  equ     casbas
00CB    datcas  equ     casbas+1

                ; **** Tastatur

0068    keybas  equ     068h
0068    keyd    equ     keybas
0069    keys    equ     keybas+1

                ; **** Bootkarte

                ; Ausgabe schaltet mit BIT7 gesetzt RAM/(EP)ROM aus
                ; mit BIT7 =0 RAM/(EP)ROM aus
                ; Bit 0..3 kann mit Ausgabe als BANK-Adresse (16..19) gesetzt werden
                ; Bit 4..6 sind derzeit unbelegt

00C8    bbport  equ     0c8h

                ; **** FLO2

                ; Belegung der Ports FDCSEL und FDCIRQ
                ;
                ; bit:   7      6       5      4       3     2     1     0
                ;fdcirq drq     irq     hld    2sided  -     -     -     -
                ;fdcsel side    moton   mini   dd      ds4   ds3   ds2   ds1

00C0    fcbas   equ     0c0h
00C0    fdccmd  equ     fcbas           ; Status und Befehlsregister
00C1    fdctrk  equ     fcbas+1         ; Trackregister
00C2    fdcsec  equ     fcbas+2         ; Sektorregister
00C3    fdcdat  equ     fcbas+3         ; Datenregister
00C4    fdcsel  equ     fcbas+4         ; Selektport
00C4    fdcirq  equ     fcbas+4

                ; FDC - Befehle
```

```

        0010    seek    equ     00010000b       ; mit HLD ohne VERIFY
        0000    home    equ     00000000b       ; mit HLD ohne VERIFY
        0088    reads   equ     10001000b       ; phys Sektor lesen IBM Format
        00A8    writes  equ     10101000b       ; phys Sektor schreiben IBM FORMAT
        00C4    readid  equ     11000100b       ; ID-Feld lesen
        00E4    readt   equ     11100100b       ; Track ( kompl ) lesen
        00F4    writet  equ     11110100b       ; Track ( kompl ) schreiben
        00D0    forcei  equ     11010000b       ; Alle laufenden Funktionen abbrechen

                        ; Bei Selekt side2-moton-minis-sdens dazufuegen soweit noetig

        0001    sela    equ     00000001b       ; Laufwerk A
        0002    selb    equ     00000010b       ; Laufwerk B
        0004    selc    equ     00000100b       ; Laufwerk C
        0008    seld    equ     00001000b       ; Laufwerk D

        0080    side2   equ     10000000b       ; Seite 2 ( Achtung JUMPER )
        0020    minis   equ     00100000b       ; BIT5=1 mini-Laufwerke
        0010    sdens   equ     00010000b       ; BIT4=1 sdens BIT4=0 ddens

                        ; ****************
                        ; XMOVE Puffer
                        ; ****************

                        ; je 'groesser' dieser Puffer ist um so schneller wird
                        ; zwischen den Banken hin und her 'geschaufelt'
                        ; 3 Dinge sind jedoch zu beachten:
                        ; Die Puffergroesse geht voll von der TPA-Groesse ab !
                        ; Der Puffer muss im gemeinsamen RAM-Bereich liegen
                        ; Eine Puffergroesse unter 100H Bytes bringt kleinen
                        ; Gewinn ... das kann MOVE vom BDOS gesteuert auch ...
                        ;
        FA00    a$bbuf  equ     0fa00h          ; Pufferstart
        FC00    a$bend  equ     0fc00h          ; Pufferende+1
        0200    a$blen  equ     a$bend-a$bbuf   ; Pufferlaenge

                        ; ****************
                        ; CCP - Bank
                        ; ****************

                        ; fuer einen schnelleren WARMBOOT, wird CCP.COM beim Kaltstart
                        ; auf Bank 0 transferiert. Von dort wird es nach jedem Warmstart
                        ; geholt anstelle es jedesmal von der Diskette zu lesen ...

        0000    ccpbnk  equ     0               ; in Bank 0
        C000    ccpis   equ     0C000h          ; dort wird CCP 'zwischengelagert'
        0D00    ccplen  equ     0d00h           ; so 'lang' ist CCP

                        ; ****************
                        ; Laufwerke
                        ; ****************

                        ; hier UNVERAENDERBARE Rechenwerte (Laufwerksbezeichnung)

                        ; einfache Dichte

        0001    maxi    equ     1               ; 8-zoll
```

```

                              ;

         0003      t40d    equ    3               ; 40 Track 2-seitig
         0005      t80d    equ    5               ; 80 Track 2-seitig

         0006      ram     equ    6               ; vorgesehen als RAM-Disk
         0007      hard    equ    7               ; vorgesehen als Harddisk

                           ; hier gewuenschter Laufwerkstyp (MAXI-T80D-T40S usw)
                           ; unter gewuenschtem Laufwerk eintragen
                           ; Nicht belegte Laufwerke muessen mit NEIN eingetragen werden

         0005      adrive  equ    t80d            ; Laufwerk A
         0005      bdrive  equ    t80d            ; Laufwerk B
         0000      cdrive  equ    nein            ; Laufwerk C
         0000      ddrive  equ    nein            ; Laufwerk D

                           ; Laufwerk E ist NUR als RAM-DISK (RAM) vorgesehen

         0000      edrive  equ    nein            ; Laufwerk E

                           ; Laufwerk F ist NUR als HARD-DISK (HARD) vorgesehen

0 Error(s) Detected.
100 Symbols Detected.
```

Cross Reference:

```
FA00  A$bBUF     208  210
FC00  A$BEND     209  210
0200  A$bLEN     210
0005  ADRIVE     246
FFFF  BANKED      50
00C8  BBPORT     156
4000  BbRAM       81
0005  BDRIVE     247
0007  BELL        59
0020  BLANK       73
0008  BS          60
00CA  CASBAS     139  140  141
0100  CCP         80
0000  CCPBNK     220
C000  CCPIS      221
0D00  CCPLEN     222
0000  CDRIVE     248
001E  CHOME       71
001A  CLS         67
00CA  CMDCAS     140
000D  CR          65
0011  CTLQ        57
0013  CTLS        58
00CB  DATCAS     141
0000  DDRIVE     249
FC36  E@CBNK     101
FC35  E@DBNK     100
FC03  ECONIN      95
FC00  ECONIST     94
FC06  ECONOUT     96
FC34  ECURBNK     99
0000  EDRIVE     253
FC27  EFLOPPY     98
001B  ESC         68
FC24  ESETBNK     97
00C0  FCBAS      166  167  168  169  170  171  172
00C0  FDCCMD     167
00C3  FDCDAT     170
00C4  FDCIRQ     172
00C2  FDCSEC     169
00C4  FDCSEL     171
00C1  FDCTRK     168
000C  FF          64
00D0  FORCEI     183
001C  FS          69
001D  GS          70
0007  HARD       240
0000  HOME       177
0009  HT          61
0049  IOCMD      111
0048  IODAT      110
0040  IOEC       109  110  111
FC00  IOSEG       83
FFFF  JA          48   49   50
0068  KEYBAS     145  146  147
0068  KEYD       146
0069  KEYS       147
000A  LF          62
```

Cross Reference:

```
0001  MAXI          232
0020  MINIS         193
F400  MONSEG         82
0000  NEIN           49   248   249   253
001F  NEWLINE        72
0000  NULL           56
0006  RAM           239
00C4  READID        180
0088  READS         178
00E4  READT         181
0031  REL            29
0030  REV            30
0010  SDENS         194
0010  SEEK          176
0001  SELA          187
0002  SELB          188
0004  SELC          189
0008  SELD          190
0080  SIDE2         192
0016  SYN            66
0003  T40D          236
0005  T80D          237   246   247
0100  TPA            79    80
00FC  UBAS          115   116
00EC  UBAS1         123   124
00CC  UBAS2         131   132   133   134   135
00FE  UCMD          118
00EE  UCMD1         126
00CE  UCMD2         134
00FF  UCON          119
00EF  UCON1         127
00CF  UCON2         135
00FC  UDAT          116   117   118   119
00EC  UDAT1         124   125   126   127
00CC  UDAT2         132
0030  USR            31
00FD  USTA          117
00ED  USTA1         125
00CD  USTA2         133
000B  VT             63
00A8  WRITES        179
00F4  WRITET        182
```

```
Z80ASM SuperFast Relocating Macro Assembler 20 Dec 85 09:46 Z80ASM 1.24 Page   1
MODEBAUDZ80 20 Dec 85 09:45

                  ;***************************************************************
                  ;*                                                             *
                  ;* rel 1.0 vom 07.01.85                                        *
                  ;*                                                             *
                  ;* Systemvereinbarungen Zeichen Ein/Ausgabe Schnittstellen     *
                  ;* und Baudrate ( falls implementiert )                        *
                  ;*                                                             *
                  ;***************************************************************

          0001    mb$input        equ   00000001b    ; EINGABE Einheit
          0002    mb$output       equ   00000010b    ; AUSGABE Einheit
          0003    mb$in$out       equ   mb$input+mb$output      ; EIN/AUSGABE Einheit
          0004    mb$soft$baud    equ   00000100b    ; Baudrate ueber Software waehlbar
          0008    mb$serial       equ   00001000b    ; Serienport evtl mit Protokoll
          0010    mb$xon$xoff     equ   00010000b    ; mit xon/xoff Protokoll

                           ;
                           ; Baudraten beziehen sich direkt auf 6551 Baustein in SER
                           ;

          0000    baud$none       equ   0            ; keine Baudrate
          0001    baud$50         equ   0001b        ; 50 baud
          0002    baud$75         equ   0010b        ; 75 baud
          0003    baud$110        equ   0011b        ; 110 baud
          0004    baud$134        equ   0100b        ; 134.5 baud
          0005    baud$150        equ   0101b        ; 150 baud
          0006    baud$300        equ   0110b        ; 300 baud
          0007    baud$600        equ   0111b        ; 600 baud
          0008    baud$1200       equ   1000b        ; 1200 baud
          0009    baud$1800       equ   1001b        ; 1800 baud
          000A    baud$2400       equ   1010b        ; 2400 baud
          000B    baud$3600       equ   1011b        ; 3600 baud
          000C    baud$4800       equ   1100b        ; 4800 baud
          000D    baud$7200       equ   1101b        ; 7200 baud
          000E    baud$9600       equ   1110b        ; 9600 baud
          000F    baud$19200      equ   1111b        ; 19.2k baud

 0 Error(s) Detected.
 22 Symbols Detected.
```

Cross Reference:

```
0003  BAUD$110          24
0008  BAUD$1200         29
0004  BAUD$134          25
0005  BAUD$150          26
0009  BAUD$1800         30
000F  BAUD$19200        36
000A  BAUD$2400         31
0006  BAUD$300          27
000B  BAUD$3600         32
000C  BAUD$4800         33
0001  BAUD$50           22
0007  BAUD$600          28
000D  BAUD$7200         34
0002  BAUD$75           23
000E  BAUD$9600         35
0000  BAUD$NONE         21
0003  MB$IN$OUT         12
0001  MB$INPUT          10    12
0002  MB$OUTPUT         11    12
0008  MB$SERIAL         14
0004  MB$SOFT$BAUD      13
0010  MB$XON$XOFF       15
```

```
                ;**************************************************************
                ;*                                                            *
                ;* Macro Definitionens fuer CP/M3 BIOS                        *
                ;*                                                            *
                ;* Aufrufvereinbarungen                                       *
                ;*                                                            *
                ;* Generierung der DRIVETABLE                                 *
                ;*                                                            *
                ;* @dtbl:        dtbl  <dph0,dph1,...> ; Name der XDPH's      *
                ;*                                                            *
                ;* Generierung der XDPH's (Parameter in einer Zeile!)         *
                ;*                                                            *
                ;* dphname:    dph     XLT-Tabelle,     ; zum Laufwerk        *
                ;*                     DPB,             ; zum Laufwerk        *
                ;*                     checksum$size,   ; (optional)          *
                ;*                     alloc$size       ; (optional           *
                ;*                                                            *
                ;* um Fehler zu vermeiden bitte die beiden letzten Parameter  *
                ;* weglassen - GENCPM.COM macht das fuer uns                  *
                ;*                                                            *
                ;* Generierung der SKEW-Tabelle                               *
                ;*                                                            *
                ;* xltname:    skew    sectors,         ; wieviel hat die Disk? *
                ;*                     skew$factor,     ; den Faktor ?        *
                ;*                     first$sector     ; Beginn mit Sektor (1) *
                ;*                                                            *
                ;* Generierung des DISK-PARTAMETER-BLOCK (dpb)                *
                ;*                                                            *
                ;* dpbname:    dpb     Sektorgroesse,   ; physikalisch        *
                ;*                     wieviel$Track,   ; vorne (+hinten)     *
                ;*                     ?Tracks,         ; pro Seite           *
                ;*                     Block$groesse,   ; in Bytes (dezimal)  *
                ;*                     DIR-Eintraege,   ; wieviel moeglich ?  *
                ;*                     Systemspuren     ; reservierte Tracks  *
                ;*                                                            *
                ;**************************************************************


                        ;
                        ; ********************************
                        ; Laufwerkstabelle (@DTBL)
                        ; ********************************
                        ;


                        ;
                        ; enthaelt immer 16 Eintraege, unbenutzte Laufwerke
                        ; werden mit 0 Eingetragen
                        ;

                dtbl    macro   ?list
                        local   ?n
                ?n      defl    0
                        irp     ?drv,<?list>
                ?n      defl    ?n+1
                        defw    ?drv
                        endm
                        rept    (16-?n)
                        defw    0
                        endm
```

```
                endm .

                ;
                ; ********************************
                ; XDPH-Macro
                ; ********************************
                ;

dph             macro   ?trans,?dpb,?csize,?asize
                local   ?csv,?alv
                defw    ?trans                  ; Adresse der SKEW-Tabelle (XLT)
                defb    0,0,0,0,0,0,0,0,0       ; BDOS Bereich
                defb    -1                      ; Media-flag
                defw    ?dpb                    ; Adresse des DPB

            if  not nul ?csize

                defw    ?csv                    ; (optional) Checksum-Vektor

            else

                defw    -2                      ; CSV wird von GENCPM eingetragen

            endif

            if  not nul ?asize

                defw    ?alv                    ; (optional) Allocation-Vektor

            else

                defw    -2                      ; ALV wird von GENCPM eingetragen

            endif

                defw    -2,-2,-2                ; DIRBCB,DTABCB,HASH und HASH-Bank
                defb    0                       ; wird von GENCPM eingetragen

            if  not nul ?csize

?csv:           defs    ?csize                  ; Checksum-Vektor

            endif

            if  not nul ?asize

?alv:           defs    ?asize                  ; Allocation-Vektor

            endif

                endm

                ;
                ; ********************************
                ; DBP-Makcro
                ; ********************************
                ;
```

```
dpb     macro   ?psize,?pspt,?trks,?bls,?ndirs,?off,?ncks
        local   ?spt,?bsh,?blm,?exm,?dsm,?drm,?al0,?all,?cks,?psh,?psm
        local   ?n

        ;;
        ;; physikalische Sektor-Mmaske und physikalischer Sektor-shift
        ;;

?psh    defl    0
?n      defl    ?psize/128
?psm    defl    ?n-1
        rept    8
?n      defl    ?n/2

    if  ?n eq 0                                 ;; = 0

        exitm

    endif

?psh    defl    ?psh+1
        endm
?spt    defl    ?pspt*(?psize/128)

?bsh    defl    3
?n      defl    ?bls/1024
        rept    8
?n      defl    ?n/2

    if  ?n eq 0                                 ;; = 0

        exitm

    endif

?bsh    defl    ?bsh+1
        endm
?blm    defl    ?bls/128-1
?size   defl    (?trks-?off)*?spt
?dsm    defl    ?size/(?bls/128)-1
?exm    defl    ?bls/1024

    if  ?dsm gt 255

    if  ?bls eq 1024

        exitm                                   ;; passt nicht mit 1k Blocks

    endif

?exm    defl    ?exm/2

    endif

?exm    defl    ?exm-1
?all    defl    0
?n      defl    (?ndirs*32+?bls-1)/?bls
        rept    ?n
```

```
?all    defl    (?all shr 1) or 8000h
        endm
?al0    defl    high ?all
?all    defl    low ?all
?drm    defl    ?ndirs-1

    if  not nul ?ncks

?cks    defl    ?ncks

    else

?cks    defl    ?ndirs/4

    endif

        defw    ?spt                    ; 128 Byte (log)-Sektoren pro Track
        defb    ?bsh,?blm               ; Block-shift und Maske
        defb    ?exm                    ; Extent-Mmaske
        defw    ?dsm                    ; Maximale Block-Anzahl
        defw    ?drm                    ; Maximale DIR-Eintraege
        defb    ?al0,?all               ; Belegungs-Vektoren
        defw    ?cks                    ; Pruefsumme
        defw    ?off                    ; Reservierte Tracks (System)
        defb    ?psh,?psm               ; (phys)-Sektor Groesse- &  Shift-Maske
        endm

        ;
        ;****************
        ; Hilfsmacros
        ; ****************
        ;

gcd     macro   ?m,?n

        ;;
        ;; Groesster gemeinsame Teiler von m,n
        ;; Ergebnis in Variabler GCDN
        ;;

?gcdm   defl     ?m                     ;; Variable  m
?gcdn   defl     ?n                     ;; Variable  n
?gcdr   defl     0                      ;; Variable  r
        rept    65535
?gcdx   defl    ?gcdm/?gcdn
?gcdr   defl    ?gcdm-?gcdx*?gcdn
        if      ?gcdr eq 0
        exitm
        endif
?gcdm   defl    ?gcdn
?gcdn   defl    ?gcdr
        endm
        endm

        ;
        ; ****************
        ; Macro fuer SKEW
        ; ****************
```

```
                ;

skew    macro   ?secs,?skf,?fsc
?nxtsec defl    0                       ;; Naechster Sektor
?nxtbas defl    0                       ;; +1 bei Ueberlauf
        gcd     %?secs,?skf

        ;;
        ;;      ?gcdn = gcd(?secs,skew)
        ;;

?neltst defl    ?secs/?gcdn

        ;;
        ;; NELTST ist die Anzahl der zu generierender Elemente
        ;; bis ein Element wiederholt wuerde
        ;;

?nelts  defl    ?neltst                 ;; Zaehler
        rept    ?secs                   ;; einmal pro Sektor
        defb    ?nxtsec+?fsc
?nxtsec defl    ?nxtsec+?skf

    if  ?nxtsec gt ?secs

?nxtsec defl    ?nxtsec-?secs

    endif

?nelts  defl    ?nelts-1

    if  ?nelts eq 0

?nxtbas defl    ?nxtbas+1
?nxtsec defl    ?nxtbas
?nelts  defl    ?neltst

    endif
        endm
        endm

0 Error(s) Detected.
5 Symbols Detected.
```

Cross Reference:

```
0000  DPB     117
0000  DPH      67
0000  DTBL     49
0000  GCD     208
0000  SKEW    235
```

Kapitel 12 Generierung der Datei CPM3.SYS

Voraussetzung ist das Vorhandensein eines CP/M Computers. Es muß sich dabei nicht notwendigerweise um einem CP/M 3 Computer handeln (praktischer wäre es aber schon).

Zur Generierung müssen folgende Programme zur Verfügung stehen:

1. Die Dateien RESBDOS3.SPR und BNKBDOS3.SPR, dies ist der von Digital Research gelieferte BDOS-Anteil an CP/M 3.

2. Ein Macro-Assembler wie Z80ASM.COM oder M80.COM.

3. Der CP/M-Linker LINK.COM (L80.COM ist nicht so ohne weiteres geeignet).

4. Das Programm GENCPM.COM das zum Lieferumfang von CP/M 3 gehört.

5. Ein Editor und evtl. ein Debugger (z.B. ZSID).

6. Alle BIOS-Dateien, wie sie in den vorangegangenen Beispielen gezeigt wurden oder die ensprechenden Gegenstücke wie sie mit einem CP/M 3 System mitgeliefert werden sollten.

   Es kann möglich sein (siehe hierzu die Computer-Unterlagen), daß einige BIOS-Segmente nur als REL-Datei geliefert wurden, und nur z.B.CHARIO wird mitgeliefert - zur Anpassung neuer Schnittstellen. Meist können alle 'restlichen' Dateien aber extra bestellt werden. (Man sollte dies tun, denn damit hat man das System besser in der 'Hand', wenn man jemals vor hat Erweiterungen vorzunehmen.)

7. Kenntnisse im Programmieren auf Assemblerebene, wenn es darum geht ein 'Fremdsystem' anzupassen.

Sind alle genannten Voraussetzungen erfüllt, die einzelnen Programm-Segmente fehlerfrei und noch genügend Platz auf der Diskette kann die Generierung der Datei CPM3.SYS beginnen:

1.Schritt:

Die Dateien SCB,BIOSKRNL,CHARIO,MOVE,DISKIO und BOOT assemblieren (oder eben die Dateien die mitgeliefert wurden). Die Zieldatei sollte eine REL-Datei sein.

2.Schritt:

Alle Dateien werden zusammengelinkt mit dem Programm LINK.COM. Dies geschieht mit folgender Syntax:

```
LINK BNKBIOS3[B]=SCB,BIOSKRNL,CHARIO,MOVE,DISKIO,BOOT
```

Mit diesem Vorgang wird die Datei BNKBIOS3.SPR generiert, der komplette BIOS-Anteil an CPM3.SYS.

3.Schritt:

Es werden nun nur noch die drei genannten SPR-Dateien und das Programm GENCPM.COM benötigt. Die SYS-Datei

wird durch einfaches Aufrufen von GENCPM generiert. Hierzu werden von GENCPM jedoch viele Fragen (leider auf Englisch) gestellt. Ein entsprechender Dialog zeigt das folgende Beispiel:

Die Übersetzung der englischen Texte wurde zur besseren Übersicht durch drei Sterne (XXX) am Beginn hervorgehoben.

Das Programm GENCPM meldet sich wie folgt:

```
CP/M 3.0 System Generation
Copyright (C) 1982, Digital Research

Default entries are shown in (parens).
Default base is Hex, precede entry with # for decimal

Use GENCPM.DAT for defaults (Y) ? _
```

XXX Die Kopfzeile benötigt kaum eine Übersetzung

darunter

XXX Vorgabewerte sind in (Klammern).

XXX Vorgabebasis ist HEX, # voranstellen für Dezimal

und gleich die erste Frage:

XXX soll GENCPM.DAT zur Vorgabe verwendet werden?

diese Frage taucht nur auf, wenn die Datei GENCPM.DAT auf der Diskette zu finden ist. Ist diese Datei vorhanden, sollte sie auf alle Fälle gesichert werden. Es folgt die Frage:

```
Create a new GENCPM.DAT file (N) ?
```

XXX soll eine neue GENCPM.DAT Datei eröffnet werden (N) ?

diese Frage muß mit 'Y' (ja) beantwortet werden, ist es nicht erwünscht, genügt ein <cr>, es kann natürlich auch ein 'N' eingegeben werden.

Eine Neueröffnung bietet den Vorteil später mit GENCPM AUTO arbeitet zu können - gleichzeitig hat man ein Protokoll.

Alle Daten werden in der Datei GENCPM.DAT unter einem Label (Namen) abgelegt, die Datei kann im Übrigen mit TYPE auf die Konsole oder den Drucker ausgegeben und/oder mit einem Editor bearbeitet werden.

Die Variablennamen werden in den nachfolgenden Beispielen immer als 'DAT-Variable' angeführt.

Wir beantworten in unserem Beispiel die Frage mit 'Y' (=ja)

Die DAT-Variable heißt CRDATAF

Es folgt die Frage:

Display Load Map at Cold Boot (Y) ?

XXX Ladewerte beim Booten ausgeben (Y) ?

mit dieser Frage ist gemeint, ob beim Kaltstart die Speicherzuweisung ausgegeben werden soll. Dies könnte wie folgt aussehen:

```
RESBIOS3 SPR  F600H  0600H
BNKBIOS3 SPR  B100H  0F00H
RESBDOS3 SPR  F000H  0600H
BNKBDOS3 SPR  8700H  2A00H
```

und gibt die Startadressen des residenten (RES) BDOS und BIOS an, also die Programmteile, auf die von allen Banken aus zugegriffen werden kann, während die ge'bank'ten Programmteile (BNK) immer auf Bank 0 zu finden sind.

Ob diese Information dem Benutzer beim Kaltstart nutzvoll ist, sei ihm selbst überlassen.

Beantwortung der Frage mit <cr> oder 'Y' bedeutet Ja, mit 'N' bedeutet Nein.

DAT-Variable: PRTMSG

Weiter geht es mit der Frage:

Number of console columns (#80) ?

XXX Anzahl der Zeichen pro Zeile (#80) ?

Diese Frage bezieht sich auf die Zeichenanzahl bei der Ausgabe von Zeichen auf die Konsole (bzw. dem Bildschirm).

Terminals mit z.B. 64 Zeichen pro Zeile können hier angepaßt werden. Terminals, die mit dem 80. Zeichen eine <cr>-<lf> Kombination ausgeben, sollten auf 79 Zeichen pro Zeile angepaßt werden.

Anmerkung:

Diese Definition bezieht sich nur auf CP/M mit seinen eigenen Unterprogrammen, nicht auf andere Programme wie z.B. Wordstar (R)

DAT-Variable PAGWID

Die nächste Frage lautet:

Number of lines in console page (#24) ?

XXX Anzahl der Zeilen pro Seite (#24) ?

Dies hängt von der verwendeten Software ab. Üblich sind 24 Zeilen. Die Statuszeile, wie sie von manchen Terminals oder Video-Kartentreibern zur Verfügung gestellt wird, zählt in diesem Zusammenhang nicht.

DAT-Variable PAGLEN

Weiter mit der Zuweisung von <backspace> und <DELete>

Backspace echoes erased character (N) ?

XXX backspace soll das letzte Zeichen wiederholen (N) ?

Üblich ist 'Nein' - <cr> oder N sonst 'Y'

DEL-Variable BACKSPC

Rubout echoes erased character (Y) ?

XXX DEL soll das letzte Zeichen wiederholen (Y) ?

Üblich ist es bei DEL(ete) das gelöschte Zeichen noch einmal auszugeben.

Dies stammt noch aus der Zeit, als die Konsolenausgabe noch nicht über Bildschirme, sondern über Drucker (oder Fernschreiber) gemacht wurde und diese konnten damals kein Backspace - wie sollte man also sonst ein erfolgtes Löschen darstellen.

Heute ist es angenehmer DEL wie Backspace zu behandeln.

<cr> oder 'Y' veranlaßt eine Zeichenwiederholung, ein 'N' behandelt DEL wie Backspace.

DEL-Variable RUBOUT

weiter mit:

Initial default drive (A:) ?

XXX welches Boot-Laufwerk (A:) ?

diese Frage ist mit <cr> zu beantworten, wenn das verwendete Monitorprogramm auf Laufwerk A bootet. Dies dürfte die übliche Bootversion sein.

DEL-Variable BOOTDRV

Es geht nun an die Zuweisung der Speicherbereiche:

Top page of memory (FF) ?

XXX Oberster RAM Bereich in Seiten -100H- (FF) ?

Bei der Zuweisung sind 3 Dinge zu beachten:

1. Das MOVE-Fenster für Programm-Transfer zwischen den Banken

2. Ein evtl. notwendiger Vektor-Bereich für Interrupts und

3. Ein evtl. vorhandener Puffer für ein VIDEO-Treiber

Das MOVE-Fenster hat am günstigsten eine Größe von 1k, da dies auch die Größe eines Sektors ist.

Fenster unter 100H bringen keinen Gewinn, denn ein 80H Fenster ist in SCB definiert und liegt sowieso im gemeinsamen RAM-Bereich.

Andernseits ist es der Wunsch vieler Benutzer einen TPA-Bereich von 60k zu haben, obwohl dies bereits mehr ist als CP/M2 in der 'normalen' single-density Version hat, denn unter CP/M 3 ist wirklich TPA gemeint, also den RAM-Bereich der dem Anwender zur Verfuegung steht. Jede extra Speicherzuweisung geht also von der TPA-Größe ab!

Achtung:

Das Programm GENCPM kann bei der Speicherzuweisung einen Rundungsfehler machen. Um dies zu umgehen ist es wichtig im letzten Programmsegment zusätzlich 100H Bytes zu reservieren.

DAT-Variable MEMTOP

Nächste Frage:

Bank-switched memory (Y) ?

XXX gebanktes RAM (Y)

Die größten Vorzüge hat CP/M3 nur im gebankten System. Sie sollten die Frage also mit <cr> oder 'Y' beantworten. Die gezeigten Assemblerprogramme sind im übrigen alle für ein gebanktes System geschrieben.

Nur wenn obige Frage mit Y beantwortet wurde, werden übrigens die nächsten Fragen gestellt:

DAT-Variable BNKSWT

Common memory base page (C0) ?

XXX Beginn des gemeinsamen RAM-Bereichs (C0) ?

Gemeint ist hier weniger der effektive Beginn des gemeinsamen RAM-Bereiches, vielmehr das mögliche ENDE des gebankten Bereiches.

Hierzu folgendes:

In der ausgelieferten Version wird der CCP in Bank 0 auf den RAM-Bereich D000-DFFF transferiert (DEFAULT.INC).

Es ist anzunehmen, daß GENCPM auch hier einen Überlauffehler hat, daher ist in der ausgelieferten Version der gemeinsame Startbereich auf C0<00>H gelegt.

Anmerkung:

Die Größe des gemeinsamen RAM-Bereichs hat nicht direkt etwas mit der Größe der TPA zu tun, bei einem gemeinsamen RAM-Bereich von 8K kann sehr wohl eine 60K TPA vorhanden sein, sind alle (nachfolgenden) Zeiger richtig gesetzt, geht lediglich in den Banks 0 und über der TPA-Bank (1) Speicherplatz verloren. Hierzu muß bei der (nachfolgenden) Definition lediglich der Speicherbereich ab E000 im Bankbereich ausgeklammert werden.

DAT-Variable COMBAS

Nächste Frage:

Long error messages (Y) ?

XXX Ausführliche Fehlermeldungen - in Englisch - (Y) ?

Auf Wunsch kann CP/M3 hier recht exakte Fehlermeldungen ausgeben, die dem Computerlaien jedoch kaum etwas sagen, ja eher zur Verwirrung beitragen. Die Entscheidung bleibt beim Anwender.

DAT-Variable LERROR

Nun wird die erste Frageserie mit:

Accept new system definition (Y) ?

XXX werden diese neuen System-Definitionen akzeptiert (Y)?

beendet.

Mit einem 'N' geht das ganze Fragespiel von vorne los, ein <cr> oder 'Y' leitet zum Ausdruck des Textes:

```
Setting up Allocation vector for drive A:
Setting up Checksum vector for drive A:
Setting up Allocation vector for drive B:
Setting up Checksum vector for drive B:

XXX Bank 1 and Common are not included XXX
XXX in the memory segment table        XXX
```

Dies deutet an, daß die Belegungsvektoren (Allocation) und die Prüfsummen-Vektoren für die Laufwerke A: und B: in Bank 0 reserviert (und belegt) wurden.

GENCPM übernimmt diese Reservierung für alle Laufwerke, die in DRVTBL zugewiesen sind und in deren XDPH (siehe dort) eine entsprechende Zuordnung durch GENCPM verlangt wird.

Es geht nun weiter mit der Speicherzuweisung in den Bankbereichen:

Number of memory segments (#3) ?

XXX Anzahl der Speicher Segmente (#3) ?

diese Frage wird nur gestellt, wenn die Frage nach dem Banking mit 'Y' beantwortet wurde.

Hier können nun die BANK's oder auch Speichersegmente eingegeben werden (bis zu 15).

Anmerkung:

Nicht mitgezählt werden darf jedoch Bank 1, da diese Bank die sogenannte TPA Bank ist (also die Bank, in der die Programme ablaufen ... sollten).

Darüberhinaus darf der Speicherbereich, der dem (gebankten) BDOS und BIOS zugeordnet ist, hier nicht nocheinmal zugewiesen werden.

GENCPM verwendet die Speichersegmente als Puffer und für die HASH-Code-Tabellen.

DAT-Variable NUMSEG

Danach folgt die Frage nach der speziellen Zuweisung, zuerst mit der Information von wo bis wo eigentlich CP/M liegt:

CP/M 3 Base,size,bank (8C,34,00)

d.h. CP/M 3 beginnt (Base) bei 8C00H, ist 3400H Bytes lang (size) und residiert auf Bank 0 (bank).

Beachten Sie bitte: 8C00H+3400H ist C000H (oder das was wir als COMBAS vorgegeben haben!).

Der Bereich 0...8C00 kann nun CP/M zugewiesen werden nach der Frage:

Enter memory segment table:

XXX Bitte Segment-Zuweisung eingeben)

Base,size,bank (00,80,00)

ein <cr> genügt, wenn der Bereich so belassen werden soll.

Wurden bei NUMSEG mehrere Segmente vorgegeben so wird nun für jedes Segment die Frage nach der entsprechenden Speicherzuweisung gestellt.

Bitte beachten: Der gemeinsame RAM-Bereich muß immer ausgeklammert bleiben (d.h. COMBAS ist immer RAM-ENDE).

DAT-Variable(n) MEMSEG00...MEMSEG0F

Ist die Zuweisung erfolgt, gibt GENCPM noch einmal die Bereich an:

```
CP/M 3 Sys     8C00H 3400H BANK 00
Memseg No. 00  0000H 8000H BANK 00
```

usw. wenn mehrere Segmente definiert wurden. Danach die Frage:

Accept new memory segment table entries (Y) ?

XXX neue Speicherzuweisung akzeptiert (Y) ?

mit <cr> gehts weiter, mit 'N' gehts von vorne los.

Setting up directory hash tables:
Enable hashing for drive a: (Y) ?

XXX Zuweisung der Hash-Tabellen :
XXX Hash Tabelle für Laufwerk A: (Y) ?

steht 'nur' eine Bank zur Verfügung kann meist nur für 2 Laufwerke eine Hash-Tabelle zugewiesen werden, diese liegt immer in Bank 0.

Die Belegung der Hashbuffer und die Zuteilung der Hash-Bank wird von GENCPM nur dann übernommen, wenn im entsprechenden XDPH (siehe später) ein FFFE eingegeben wird.

DAT-Variable HASHDRVA....HASHDRVP (Laufwerk A bis P)

Danach werden wieder Informationen ausgegeben:

Setting up Blocking/Deblocking Buffer
The physikal record size is 400H

Available space in 256 byte pages
TPA = 00F0H, Bank 0 = 0078H, other banks = 0000H

XXX Belegung des Blocking/Deblocking-Puffers
XXX die physikalische Blockgröße ist 400H (das ist 1K)

XXX Verfügbarer Speicherplatz in 256 Byte Seiten
XXX TPA = 00F0H, Bank 0 = 0078H, andere Banks = 0000H

Die nachfolgenden Fragen sind nun recht unterschiedlich zu beantworten, mit entsprechend unterschiedlichen Ergebnissen.

Es kann die Größe von zwei verschiedenen Puffern vorgegeben werden, die im gebankten Bereich liegen. In diese Puffer werden zum einen die Directory und zum andern Daten eingelesen.

Je größer die Puffer definiert werden, umso mehr (und umso schneller) kann auf einmal eingelesen werden. Bei nur einer Bank ist die Kapazität natürlich schnell erschöpft.

Zu beachten ist darüberhinaus, daß der Pufferbereich für das Inhaltsverzeichnis NUR in Bank 0 angelegt wird, bei vielen Laufwerken kann es da schon einmal 'eng' werden.

Die ersten Fragen betreffen Laufwerk 'A', dieses sollte u.U etwas bevorzugt 'behandelt' werden, da dieses Laufwerk am meisten benutzt wird:

Number vor directory buffers for drive A:(#1) ?

XXX Anzahl der DIR-Puffer für Laufwerk A:(#1) ?

Die Frage kann mit bei 2 Laufwerken mit #8 beantwortet werden, beachten Sie aber bitte die jeweils folgenden Angaben über den 'restlichen' Speicherplatz.

DAT-Variable NDIRRECA...NDIRRECP

Es folgt: (nach der Restspeicherangabe, wie oben übersetzt)

Number of data buffers for drive A:(#1) ?

XXX Anzahl der Daten-Puffer für LW A:(#1) ?

Auch diese Frage kann mit #8 beantwortet werden.

DAT-Variable NDTARECA...NDTARECP

gleich danach wird gefragt:

Allocate buffers outside of Common (Y) ?

XXX Allocation-Puffer ausserhalbe des gemeinsamen Bereiches verlegen (Y) ?

Diese Frage beantworten Sie bitte mit 'Y', sonst wird nur die TPA unnötig verkleinert.

Im anderen Falle sind die Allocation-Vektoren zwar aus der TPA heraus zwar zugänglich, aber da dies nicht immer der Fall sein muß, nützt diese Kenntnis eigentlich wenig, ausser für Sonderanwendungen.

DAT-Variable ALTBNKSA...ALTBNKSP

Diese 'Fragerei' geht nun für alle eingebundenen Laufwerke weiter.

Werden Fehler bei der Belegung gemacht, so meldet GENCPM dieses mit:

ERROR: unable to allocate .... buffer space

XXX FEHLER: kann ...... Pufferplatz nicht zuweisen

wobei anstelle '......' angegeben wird, ob es DIR oder Datenpuffer ist, der nicht zugewiesen werden kann.

Ist alles zugewiesen und von GENCPM akzeptiert, folgt die Frage:

Accept new puffer definitions (Y) ?

XXX ist neue Pufferdefinition akzeptiert (Y) ?

Sind Sie damit zufrieden genügt ein <cr>, wollen Sie noch weitere Möglichkeiten probieren geben Sie ein 'N' ein.

Haben Sie die Frage mit 'Y' oder <cr> beantwortet gibt GENCPM die gültige Speicheraufteilung aus:

```
BNKBIOS3 SPR   F600H 0800H    (also von F600H..FDFFH )
BNKBIOS3 SPR   B800H 0800H    (also von B800H..BFFFH )
RESBDOS3 SPR   F000H 0600H    (also von F000H..F5FFH )
BNKBDOS3 SPR   8B00H 2D00H
```

XXX CP/M 3.0 SYSTEM GENERATION DONE XXX

XXX CP/M 3.0 System Generierung abgeschlossen

Der Vollständigkeit halber, ist nachfolgend nocheinmal ein kompletter Durchlauf (mit PUT aufgezeichnet) abgedruckt.

```
B>gencpm


CP/M 3.0 System Generation
Copyright (C) 1982, Digital Research

Default entries are shown in (parens).
Default base is Hex, precede entry with # for decimal

Use GENCPM.DAT for defaults (Y) ? n

Create a new GENCPM.DAT file (N) ? y

Display Load Map at Cold Boot (Y) ? y

Number of console columns (#80) ? #80
Number of lines in console page (#24) ? #24
Backspace echoes erased character (N) ? n
Rubout echoes erased character (Y) ? n

Initial default drive (A:) ? <cr>

Top page of memory (FF) ? fa
Bank switched memory (Y) ? y
Common memory base page (C0) ? c0

Long error messages (Y) ? y

Accept new system definition (Y) ? y

Setting up Allocation vector for drive A:
Setting up Checksum vector for drive A:
Setting up Allocation vector for drive B:
Setting up Checksum vector for drive B:

*** Bank 1 and Common are not included ***
*** in the memory segment table.       ***

Number of memory segments (#3) ? 1

CP/M 3 Base,size,bank (8D,33,00)

Enter memory segment table:
 Base,size,bank (00,80,00) ?<cr>

 CP/M 3 Sys    8D00H 3300H  Bank 00
 Memseg No. 00 0000H 8000H  Bank 00

Accept new memory segment table entries (Y) ? y

Setting up directory hash tables:
 Enable hashing for drive A: (Y) ? y
 Enable hashing for drive B: (Y) ? y
```

```
The physical record size is 0400H:

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0078H, Other banks = 0000H

               Number of directory buffers for drive A: (#1) ? 8

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0057H, Other banks = 0000H

               Number of data buffers for drive A: (#1) ? 8
               Allocate buffers outside of Common (N) ? y

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0037H, Other banks = 0000H

               Number of directory buffers for drive B: (#1) ? 8

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0016H, Other banks = 0000H

               Number of data buffers for drive B: (#1) ? 8
               Allocate buffers outside of Common (N) ? y

ERROR:  Unable to allocate Data deblocking buffer space.

**** da passte wohl was nicht - das Menue beginnt von neuem ***

The physical record size is 0400H:

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0078H, Other banks = 0000H

               Number of directory buffers for drive A: (#8) ? 8

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0057H, Other banks = 0000H

               Number of data buffers for drive A: (#8) ? 8
               Allocate buffers outside of Common (Y) ? y

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0037H, Other banks = 0000H

               Number of directory buffers for drive B: (#8) ? 4

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0026H, Other banks = 0000H

               Number of data buffers for drive B: (#8) ? 4
               Allocate buffers outside of Common (Y) ? y

     Available space in 256 byte pages:
     TPA = 00EFH, Bank 0 = 0016H, Other banks = 0000H
```

```
Accept new buffer definitions (Y) ? y

 BNKBIOS3 SPR  F500H  0600H
 BNKBIOS3 SPR  BA00H  0600H
 RESBDOS3 SPR  EF00H  0600H
 BNKBDOS3 SPR  8C00H  2E00H

*** CP/M 3.0 SYSTEM GENERATION DONE ***
```

TEIL V Einblicke, Besonderheiten und Ausblicke

Kapitel 1 Einblicke auf die Diskette

Es wurde in den vorangegangenen Kapiteln viel über Inhaltsverzeichnisse gesprochen und über Tracks, Sektoren usw. Es ist an der Zeit, sich einmal im wahrsten Sinne des Wortes ein Bild davon zu machen wie das den wohl 'wirklich' aussieht.

Das nachfolgende Beispiel zeigt einmal den Inhalt der ersten 8 Zeilen eines Inhaltsverzeichnisses.

```
00: 0053 5441 544C 494E 4549 4E43 0000 0045  .STATLINEINC...E
10: 1000 1100 1200 1300 1400 0000 0000 0000  ................
20: 0056 3933 3636 2020 2052 454C 0000 0007  .V9366   REL....
30: 1500 0000 0000 0000 0000 0000 0000 0000  ................
40: E552 4543 414C 4C20 205A 3830 0000 0020  .RECALL  Z80...
50: 1700 1800 0000 0000 0000 0000 0000 0000  ................
60: 0052 4543 414C 4C20 2043 CF4D 0000 0003  .RECALL  C.M....
70: 3101 0000 0000 0000 0000 0000 0000 0000  1...............
```

Acht Zeilen wurden deshalb gewählt, weil sie gerade dem Inhalt von einem logischen Sektor (128-Bytes=80H-Bytes) entsprechen.

In diese Sektorgröße passen 4 DIR-Einträge falls, das Inhaltsverzeichnis nicht mit INITDIR 'vorbehandelt' wurde, andernfalls nur 3 Einträge.

Das Beispiel soll nun einmal näher untersucht werden:

Die jeweils ersten 3 Spalten sind relativ uninteressant, sie stellen lediglich die relative Adresse des ersten Bytes nach dem Doppelpunkt dar. Die jeweils folgenden Bytes entsprechen demnach den (relativen) Adressen 01,02 ... 11,12 bis 7F. Alle Angaben sind übrigens in HEX (also Basis 16).

Jeweils 2 Zeilen (also von 00...1f, 20..3f usw.) entsprechen einem FCB (File Control Block). In diesem FCB sind alle Angaben enthalten, die das BDOS über eine bestimmte Datei wissen muß.

Diese Angaben sollen nun nachfolgend analysiert werden. Beginnen wir mit der ersten Datei:

Byte 00 hier sind folgende Einträge möglich:

00..0F	Dies gibt die USER-Ebene an, in welcher die Datei abgespeichert wurde. (00=0,01=1 usw)
20	Gibt an, daß nachfolgend der Diskettenname eingetragen ist. (nur in einem erweiterten Inhaltsverzeichnis möglich)
21	Bezeichnet den Zusatz-FCB für jeweils 3 DIR-Einträge in einem erweiterten Inhaltsverzeichnis.
E5	Bezeichnet eine gelöschte Datei.

Byte 01..08 Ab Byte 1 folgt der Dateiname. Dieser muß immer in Großbuchstaben eingetragen sein.

Der Grund liegt darin, daß der CCP grundsätzlich alle Eingaben vor der eigentlichen Abarbeitung in Großbuchstaben umwandelt. Ein Dateiname in Kleinbuchstaben könnte also niemals aufgerufen werden, er könnte nicht einmal (so ohne weiteres) aus dem Inhaltsverzeichnis gelöscht werden.

Der Dateiname darf bis zu acht Buchstaben lang sein. Ist der Name kürzer, muß der Rest mit Leerzeichen (20H) aufgefüllt werden.

Byte 09..0B — Es folgt der Index. Er darf 3 Zeichen lang sein, im übrigen gilt auch hier das für den Dateinamen gesagte.

Byte 0C — Dies ist das sog. EX(tend)-Feld. Ist eine Datei größer als 16k, wird für jeweils weitere 16K ein sog. EX(tend) angelegt. Darunter ist ein weiterer FCB mit gleichem Dateinamen zu verstehen, der die jeweils nächsten 16K 'verwaltet'.

Es sind bis zu 32 EXtends möglich (00..1f). Die entsprechende EXtend-Nummer wird in diesem Feld abgelegt. Dies bedeutet im übrigen, daß unter CP/M 3 die maximale Größe einer Datei 512K betragen kann. (32X16K)

Byte 0D..0E — Diese Felder werden vom BDOS für interne Zwecke benötigt. Sie müssen im Inhaltsverzeichnis auf NULL gesetzt sein.

Byte 0F — ist das sog. RC-Feld (record-count).Darunter ist ein Zähler zu verstehen, der genauere Angaben über die Dateigröße macht, die Zahl bedeutet nämlich nichts anderes als die Dateigröße in logischen (128-Byte) Sektoren.

Es sind Einträge von 00..80H möglich. 00 würde einen Leer-FCB bezeichnen (also nur ein Namenseintrag, aber keine abgelegten Daten) und 80H würde die max. mögliche Dateigröße eines EXtends bezeichnen.

(16K=128(80H)X128(log Sektor).

Die Angaben in den Bytes 00..0F entsprechen im wesentlichen dem, was zum Aufbau eines FCB's in früheren Kapiteln gesagt wurde. Lediglich Byte 00 macht hier eine Ausnahme - nicht die Laufwerksnummer ist hier eingetragen, sondern .... (siehe oben).

Die Sache mit dem Eintragen der Laufwerksnummer in dieses Byte innerhalb eines FCB's wird sowieso etwas unüblich gehandhabt, mal bedeutet 00 das gerade 'eingeloggte' Laufwerk und mal Lauf-werk 'A', aber wie bei allem was mit Software zu tun hat, man soll so etwas nicht zu 'verkniffen' sehen, wichtig ist haupt-sächlich, daß man weiß, was innerhalb eines Programmes gerade gemeint ist.

Die zweite Zeile des FCB's (Bytes 10..1F) kann nun schon einige Rätsel aufgeben, wenn man nicht weiß, wo es dabei 'lang' geht. Es

ist eigentlich ganz einfach, jeweils zwei Bytes bezeichnen eine fortlaufende Blocknummer, in welcher ein Datenblock zu finden ist.

Doch nun eine etwas genauere Analyse:

Zuerst, was ist ein Datenblock: darunter ist eine im DPB für jedes (logische) Laufwerk festgelegte Datengröße zu verstehen. Der Wert läßt sich aus BSH oder BLM erkennen. (siehe Teil IV Kapitel 9 DISKIO).

Im gezeigten Beispiel ist eine Blockgröße von 2048 Bytes (2K) vorgegeben (das muß man wissen - kann aber leicht mit SHOW in Erfahrung gebracht werden).

Nun muß man nur noch wissen, daß der erste Datenblock (0000) immer mit dem ersten Sektor des Inhaltsverzeichnis beginnt. Das Inhaltsverzeichnis seinerseits ist der erste Sektor auf dem ersten Track nach den sog. Systemspuren.

Die Systemspuren ihrerseits sind im DPB wiederum als OFF angegeben, als OFFset von Track 0 (immer in ganzen Track - keinesfalls in Sektoren).

Mit diesem Wissen und der Kenntnis, daß CP/M NUR logische Sektoren kennt (außer im BIOS) wird die Sache ganz einfach:

> Die Blocknummer gibt den Offset in Blöcken vom ersten Eintrag im Inhaltsverzeichnis an. Blöcke sind normalerweise 1024, 2048 oder 4096 Bytes groß (bei Harddisks auch 8 bis 16K). Die Blockgröße entspricht dem Faktor (BLM+1)*128 Bytes. (BLM+1*logischer Sektorgröße).

Bevor wir die Geheimnisse der Blocknummern nun ganz genau betrachten, zuerst einmal die Angaben von SHOW zu dem Laufwerk aus welchem der Auszug des Inhaltsverzeichnisses stammt:

```
      F: Drive Characteristics
 81,920: 128 Byte Record Capacity
 10,240: Kilobyte Drive  Capacity
  1,024: 32 Byte  Directory Entries
      0: Checked  Directory Entries
    128: Records / Directory Entry
     16: Records / Block
    256: Sectors / Track
      0: Reserved  Tracks
    256: Bytes / Physical Record
```

Die Angabe 10,240 Kilobyte Drive Capacity (Laufwerkskapazität) läßt leicht darauf schließen, daß es sich hier um eine Harddisk handelt, Floppys haben (leider noch) keine solchen Speicherkapazitäten.

Es ist aus den Daten ebenfalls zu entnehmen, daß die Blockgröße (16 Records/Block=16*128Bytes=) 2048 Bytes beträgt (wobei jeder physikalische Sektor 256 Bytes=2 logische Sektoren groß ist).

Die letzte wichtige Eintragung ist, daß 0 reservierte Tracks vorhanden sind, das Inhaltsverzeichnis beginnt also in Sektor 1 Track 0.

Betrachten wir nun die erste Blocknummer im obigen Beispiel finden wir dort die Zahl 1000. Nun computertypisch für 8080 und Z80-CPU's sind diese Angaben (in HEXadezimal) im Format -unterer Wert (LOW Byte) -oberer Wert (HIGH Byte), also mit der Bedeutung 0100=01H, 1000=10H, 0001=0100H und 0010=1000H.

Dies ist sicher etwas gewöhnungsbedürftig hängt aber damit zusammen, daß in diesem Format Doppelregisterinhalte abgelegt und zurückgelesen werden.

Fangen wir nun einmal an zu rechnen: 10H=16 dezimal. 16*Block-Größe=(16*2048)=32768=32K. Nun betrachten wir einmal die mit SHOW festgestellte Anzahl der DIR-Einträge (1,024 32 Bytes Directory Entries). Wir stellen fest, daß die angegeben Blocknummer genau auf den ersten Block hinter dem Inhaltsverzeichnis zeigt.

Um den ersten Sektor zu finden, in welchem sich die Datei STATLINE.INC befindet, sind nun nur noch wenige Berechnungen notwendig. (CP/M kann das übrigens viel schneller als wir mit dem Taschenrechner oder gar aus dem Kopf).

Rechnen wir erst einmal um, wieviele logische Sektoren 32768 Bytes sind, es sind genau (32768/128=) 256 logische (128 Byte) Sektoren, soviel logische Sektoren wie genau auf einen Track passen. Die gesuchte Datei beginnt also in Sektor 1 Track 1.

Um den ersten Block zu lesen, müssen wir (2048/128=)16 logische Sektoren lesen. Damit ist die Datei aber noch nicht völlig gelesen; es stehen noch mehr Angaben im FCB, nämlich die Nummer des zweiten Blockes. Er läßt sich genauso berechnen wie oben gezeigt.

Da es gerade der nächste Block ist, kann man sich die Angelegenheit natürlich erleichtern, er beginnt mit Sektor 17 (1+16).

Fassen wir dies alles noch einmal zusammen:

> Eine Datei beginnt dort, wohin die erste Blockadresse zeigt.
>
> Die Dateigröße läßt sich aus dem EX-Feld und dem RC-Feld erkennen. Ist das EX-Feld nicht 00 sind noch weitere FCB's zu der entsprechenden Datei vorhanden. Das RC-Feld gibt in jedem Falle an, aus wievielen logischen Sektoren die Datei besteht.
>
> Aus der Blockadresse ist der logische Sektor und der Track zu errechnen. Die Berechnungsgrundlage hierfür lautet:
>
> Gesamtsektoren=(Blocknummer*(BLM+1)
>
> Track=(Gesamtsektoren/SPT)+OFF (Rest entfällt) siehe DPH
>
> Logischer Sektor=Gesamtsektoren-Track(ohne Rest)*SPT

Beispiel:

Berechnen des ersten (logischen) Sektors der Datei V9366.REL im gezeigten Inhaltsverzeichnis:

Gesamtsektoren=21(15H)*16=336

Track=(336/256)+0=1,..

Logischer Sektor=336-(1*256)=80

Der physikalische Sektor in welchem sich der logische Sektor befindet, ist (ebenfalls mit Daten aus dem DPB) zu berechnen mit

Physikalischer Sektor=(Log.Sektor/(PHM+1))

Ein eventueller Rest deutet an, daß es sich um einen logischen Sektor handelt, der 'innerhalb' des gefunden physikalischen Sektors liegt. (In einem 256 Byte Sektor sind ja zwei logische Sektoren enthalten.

Der gefundene Wert geht davon aus, daß ein Track mit Sektor 0 beginnt, dies ist unter CP/M 3 zwar möglich, es hat sich aber bereits unter CP/M 2 eingebürgert einen Track mit Sektor 1 zu beginnen und nicht wie bei der Track-Nummerierung mit 0. Dem gefunden Wert ist also eine 1 zuzufügen.

Demnach startet die Datei V9366.REL auf Track 1 und dem physikalischen Sektor 41.

Hier kann noch eine weitere Falle eingebaut sein, nämlich der SKEW-Faktor.

Ist dies der Fall, hilft es nur noch (über die BDOS-Funktion 50) die BIOS-Funktion SECTRN aufzurufen. (Siehe Teil IV Kapitel 9 DISKIO).

Anmerkung:

Die meisten Disketten-Inspektionsprogramme 'arbeiten' mit der Angabe der logischen Sektornummer, soll aber auf einen bestimmten Sektor über das BIOS zugegriffen werden, so ist immer mit dem physikalischen Sektor zu rechnen.

Bei zweiseitigen Disketten kann darüber hinaus die Angabe des OFFset (oder anders ausgedrückt die Anzahl der reservierten Systemspuren) etwas irreführend sein. Dies ist der Fall, wenn, wie in der vorliegenden BIOS-Beschreibung, die Seitenumschaltung aus der Tracknummer abgeleitet wird; in diesem Fall wird eine Diskette logisch so aufgeteilt, als hätte sie die doppelte Anzahl von Tracks, nämlich z.B. 80 Track auf Seite 0 und 80 Tracks auf Seite 1, dafür aber auch nur soviel Tracks wie wirklich pro Seite vorhanden sind.

Es kann hier natürlich auch mit einem Offset von einer Systemspur 'gearbeitet' werden, in den meisten Fällen wird dies jedoch vermieden, um damit zu gewährleisten, daß das Inhaltsverzeichnis immer auf Seite 0 einer Diskette beginnt.

Ein weiteres Beispiel soll noch die 'Eigenarten' des erweiterten Inhaltsverzeichnisses verdeutlichen, wie es mit INITDIR generiert werden kann:

```
00: 2040 4153 5445 5220 2030 3031 3100 0000   MASTER 0011
10: 0000 0000 0000 0000 400B 1021 820B 1113  ........@.!.....
20: 0053 4342 2020 2020 2052 454C 0000 0003   SCB     REL....
30: 4200 0000 0000 0000 0000 0000 0000 0000  B...............
40: 0044 4953 4B49 4F20 2052 454C 0000 0013   DISKIO  REL....
50: 0202 0302 0000 0000 0000 0000 0000 0000  ................
60: 2140 0B10 2182 0B11 1300 0072 0B15 2372  !@..!......r..#r
70: 0B15 2300 007F 0B12 407F 0B12 4000 0000  ..#.....@...@...
```

Auf den ersten Blick sind die DIR-Einträge kaum vom voran gegangenen Beispiel zu unterscheiden. Erst beim zweiten Blick findet man gewisse Unterschiede:

Der erste Eintrag beginnt mit 20, hier handelt es sich also um einen LABEL-Eintrag, d.h. um den Namen der Diskette.

Das EX-Feld wird hier zu anderen Dingen 'mißbraucht'. Hier hat jedes gesetzte (=1) Bit eine besondere Bedeutung:

Bit 7	Die Diskette ist Paßwortgeschützt
Bit 6	Zeit- und Datumsmarken setzen beim Dateizugriff (access)
Bit 5	Zeit- und Datumsmarken setzen bei Dateierneuerung (update)
Bit 4	Zeit- und Datumsmarken setzen bei Neueröffnung einer Datei (create)
Bit 3	Ein Diskettenname (Label) wurde zugewiesen.

Byte 10..17	hier steht (in verschlüsselter Form) das Paßwort, das der Diskette zugewiesen wurde.
Byte 18..19	Hier steht das Datum, an welchem der Diskettenname zugewiesen wurde (create).
Byte 1A..1B	Hier steht die dazugehörende Uhrzeit (in BCD)
Byte 1C..1D	Hier steht das Datum an welchem der Diskettenname zuletzt geändert wurde
Byte 1E..1F	Hier steht die dazugehörende Uhrzeit (in BCD)
	Das Datum ist immer als Offset (in HEX) vom 1.1.1978 eingegeben. Die Uhrzeit nur mit Stunden (erstes Byte) und Minuten (zweites Byte).

Ab der relativen Adresse 60 beginnt das zusätzliche Datenfeld für die drei vorangegangenen Dateien.

Dieses Datenfeld ist wie folgt aufgebaut:

Byte 00	21 = Kenner

Byte 01..04	Datums- und Zeitmarke (create oder access) für die erste Datei im Feld.
Byte 05..08	Datums- und Zeitmarke (update) für die erste Datei im Feld.
Byte 09	Paßwortmodus (siehe BDOS-Funktion 102)
Byte 0A	reserviert vom Betriebssystem.
Byte 0B..0E	Datums- und Zeitmarken (create oder access) für die zweite Datei im Feld
Byte 0F..12	Marken für update der zweiten Datei
Byte 13	Paßwortmodus der zweiten Datei
Byte 14	reserviert für die zweite Datei
Byte 15..18	Marken (create oder access) der dritten Datei
Byte 19..1C	Marken (update) der dritten Datei
Byte 1D	Paßwortmodus der dritten Datei
Byte 1E	reserviert für die dritte Datei
Byte 1F	frei (?)

Aus diesen Angaben kann recht einfach erkannt werden, daß z.B. die Datei SCB.REL um 15.23 Uhr 'tätig' war.

Das Datum ist nicht so ohne weiteres feststellbar, es sind halt 0B15H (=2837) Tage seit dem 1.1.1978 vergangen, (grob 7,77 Jahre also etwa Juli/ August 1985).

Es gibt zur Berechnung des genauen Datums und des Wochentages bestimmte Algorithmen deren Erläuterung jedoch den Rahmen dieses Buches sprengen würden.

Kapitel 2 ... nun noch das Besondere - Interrupts und RAM-Floppy

In den vorangegangenen Kapitels wurden (fast) alle Details des BDOS und des BIOS erwähnt, aber ganz sicher ist dem Systemprogrammierer mit den hier gegebenen Unterlagen immer noch nicht gedient. Leute dieser Art - und ich zähle mich selbst dazu - sind mit allen Unterlagen die sie bekommen können, unzufrieden es könnte doch irgendwo noch ein kleines Geheimnis stecken, das ausgenutzt werden kann ...

Sicher würde hier das Listing des CCP und des BDOS ein ganzes Stück weiter helfen, aber diese sind eben nicht lieferbar, wenn es auch bereits eine ganze Reihe von Quellen für disassemblierte und nachkommentierte Listings gibt. Inwieweit jedoch derartige 'Unterlagen' von Wert sind hängt sicher von der Qualität der 'Übersetzung' ab und vom jeweiligen 'Verwendungszweck'.

Totzdem gibt es einige Besonderheiten, die nirgends in den Originalunterlagen besprochen werden, dem Betroffenen aber das Leben doch recht schwer machen können.

Eines dieser Dinge ist die Anwendung von Interrupts unter CP/M3.

Der Interrupt 'lebt' davon, daß bei seinem Auftreten ein bestimmter Vektor auf die sog. Interrupt-Service-Routine zur Verfügung steht. So weit so gut, aber 8-Bit Prozessoren können im allgemeinen nur einen Speicherbereich von 64k adressieren, aus diesem Grunde wurde für CP/M 3 ja auch das Banking eingeführt, um mehr Speicherplatz adressieren zu können. Wenn nun ein Interrupt auftritt, steht der CPU zwar die Adresse der Service-Routine zur Verfügung aber ob diese Routine auch gerade in der Bank steht die gerade eingeschaltet ist?

Hier bieten sich mehrere Lösungen an:

1. Im Interrupt Modus 0 (8080 Mode)

   Hier kann die unterste Seite (Adresse 0..FFH) in Bank 0 und allen Banken oberhalb Bank 1 reserviert werden (geht mit GENCPM). Die Sprungvektoren müssen dann aber auf allen (möglichen) Banken eingetragen werden. (Natürlich auch auf Bank 1 !).

   Zeigen müssen die Vektoren jedoch in jedem Falle in den gemeinsamen Speicherbereich, denn nur dieser ist aus allen Banken erreichbar.

2. Im Interrupt Modus 1 (RST 38)

   Hier gilt das unter (1) gesagte. In Bank 1, der TPA-Bank sind jedoch besondere Vorkehrungen zu treffen, da die Adresse 0038H auch von SID & Co verwendet wird, dies galt aber auch schon für CP/M 2.

3. Im Interrupt Modus 2 (Vector Interrupt)

   Dieser Modus ist natürlich nur mit Z80-Prozessoren und entsprechender Periferie möglich. Es ist jedoch die einfachste Lösung für viele Probleme.

Das nachfolgende Listing zeigt die Anwendung eines Z80-CTC's als Interrupt-Geber für eine software-Uhr unter CP/M 3. Die Initialisierung erfolgt mit Aufruf des DATE-Befehles.

Z80ASM SuperFast Relocating Macro Assembler
STIME Z80 25 Jan 86 11:58

TIME.TXT

```
                                MACLIB  DEFAULT.INC
                        ; **************************************************************
                        ; *                                                            *
                        ; * MODUL       TIMESOFT.INC                                   *
                        ; * REL         2.0                     850625                 *
                        ; *                                                            *
                        ; * Dieses Programmsegment unterstuetzt eine software-Uhr mit  *
                        ; * dem Zilog-Baustein CTC.                                    *
                        ; *                                                            *
                        ; * Die software-Uhr laeuft im Sekunden-interrupt und ist voll *
                        ; * CP/M3 kompatibel ( d.H. die Zeit wird mit DATE gestellt -  *
                        ; * laeuft aber erst NACH dem Stellen an - und kann mit DATE   *
                        ; * oder einem entsprechenden BDOS-Call gelesen werden         *
                        ; *                                                            *
                        ; * Wird die Zeit in der Statuszeile benutzt werden keine      *
                        ; * Sekunden angezeigt.                                        *
                        ; *                                                            *
                        ; * geschrieben von RAOUL O. KOERBER, Detmod                   *
                        ; *              fuer ELZET-80 Mikrocomputer GmbH & CO KG      *
                        ; *                                                            *
                        ; **************************************************************


                                ;
                                ; Diese Datei muss in die Datei BOOT eingebunden werden
                                ; (mit einem INCLUDE-Befehl)
                                ; Die nachfolgend genannten externen Variablen sind dann
                                ; zu entfernen - sie dienen lediglich dazu diese Datei
                                ; als Muster zu assemblieren.
                                ;

                                extrn   @date,@sec      ; im SCB

                                cseg

                                ;
                                ; ***************
                                ; * Zeit/Datum  *
                                ; ***************
                                ;

0000'                   ?time:

                                ;
                                ; Aufruf von ?TIME erfolgt mit <C>=FF wenn die Zeit
                                ; gestellt werden soll. Die einzustellende Zeit uebergibt
                                ; CP/M3 im SCB ( siehe dort )
                                ; Bei <C>=0 soll die Zeit im SCB auf den neuesten Stand
                                ; gebracht werden. Da die CTC-Uhr im Interrupt betrieben
                                ; wird, ist diese Funktion ueberfluessig - eine Auffrischung
                                ; erfolgt im Sekundentakt
                                ;

```

```
0000' 0C                 inc     c               ; Initialisieren?
0001' C0                 ret     nz              ; ... fertig wenn nicht


                         ;
                         ; ****************
                         ; * Einstellen   *
                         ; ****************
                         ;

                         ;
                         ; beim Zeitsetzen wird lediglich der CTC initialisiert
                         ; die Zeit hat CP/M bereits im SCB abgelegt
                         ;

0002'            settime:

                         ;
                         ; initialisieren des CTC
                         ;

0002' F3                 di                      ; keine Unterbrechung
0003' E5                 push    hl
0004' D5                 push    de
0005' 3E FF              ld      a,high ivect0
0007' ED 47              ld      i,a
0009' ED 5E              im      2
000B' 21 0038'           ld      hl,ctcserv
000E' 22 FF02            ld      (ivect1),hl     ; Vektor
0011' 21 0028'           ld      hl,clkinit
0014' CD 001B'           call    send$init
0017' D1                 pop     de
0018' E1                 pop     hl
0019' FB                 ei
001A' C9                 ret

001B'            send$init:

                         ;
                         ; Ausgabe eines Initialisierungs-Strings.
                         ; 1. Zeichen im String ist die Anzahl der zu sendenden
                         ; Zweiergruppen. Jede Zweiergruppe besteht aus
                         ; 1. Einer Portadresse  2. Des Initialisierungswertes
                         ;

001B' 46                 ld      b,(hl)          ; Laenge des INITstrings(/2)
001C' 04                 inc     b
001D' 05                 dec     b               ; evtl 0 = keine Init notwendig ?
001E' C8                 ret     z
001F' 23         sndini1:inc     hl              ; naechster Wert
0020' 4E                 ld      c,(hl)          ; Initport
0021' 23                 inc     hl              ; naechster Wert
0022' 7E                 ld      a,(hl)          ; Initwert
0023' ED 79              out     (c),a           ; Wert senden
0025' 10 F8              djnz    sndini1         ; fertig?
0027' C9                 ret
```

```
0028' 07          cikinit:db      7               ; Stringlaenge/2
0029' 20 03               db      p$ctc0,resctc
002B' 21 03               db      p$ctc1,resctc
002D' 20 00               db      p$ctc0, low ivect0
002F' 20 35               db      p$ctc0,timer
0031' 20 7D               db      p$ctc0,tim40
0033' 21 C5               db      p$ctc1,counter
0035' 21 7D               db      p$ctc1,cntcon

0037' 00          usrflg: db      0               ; NZ nach Interrupt

                          ;
                          ; ****************
                          ; * INT-Service  *
                          ; ****************
                          ;

                          ;
                          ; Anmerkung betreffend Interrupts unter CP/M3:
                          ;
                          ; Es ist NUR Mode 2 sicher, da der Interrupt prinzipiell auf
                          ; jeder Bank moeglich sein muss. Der Interrupt-Vektor ist ab
                          ; Adresse 0FF10H einzutragen (0FF00H...0FF0FH ist fuer System-
                          ; software - z.B. die UHR ) freizuhalten. Die Interuptservice-
                          ; Routine muss innerhalb des gemeinsamen (common) Bereiches liegen.
                          ; Sie kann entweder unter CSEG in das BIOS eingebunden werden oder
                          ; (falls kurz genug) innerhalb dem Vektor-Bereich von 0FF10H ...
                          ; 0FF7FH liegen. (der Bereich dahinter ist wegen u.U. schlechter
                          ; Stackmodalitaeten einiger Programme nicht sicher !)
                          ; Bitte Beachten: das LOW-Byte des Interruptvektors muss immer auf
                          ; einer geraden Speicheradresse liegen ....
                          ;

0038' F3          ctcserv:di
0039' F5                  push    af
003A' E5                  push    hl
003B' 21 0000#            ld      hl,@sec
003E' 7E                  ld      a,(hl)          ; lese Sekunden
003F' C6 01               add     a,1
0041' 27                  daa
0042' 77                  ld      (hl),a          ; Sekunden + 1
0043' FE 60               cp      60h             ; ganze Minute ?
0045' 38 24               jr      c,exit1         ; nein

0047' 36 00               ld      (hl),0          ; reset Sekunden
0049' 2B                  dec     hl              ; --> Minuten
004A' 7E                  ld      a,(hl)          ; und lesen
004B' C6 01               add     a,1
004D' 27                  daa
004E' 77                  ld      (hl),a          ; Minuten + 1
004F' FE 60               cp      60h             ; ganze Stunde ?
0051' 38 15               jr      c,exit          ; wenn nein

0053' 36 00               ld      (hl),0          ; reset Minuten
0055' 2B                  dec     hl              ; --> Stunden
0056' 7E                  ld      a,(hl)          ; und lesen
0057' C6 01               add     a,1
0059' 27                  daa
005A' 77                  ld      (hl),a          ; Stunden + 1
005B' FE 24               cp      24h             ; ganzer Tag ?
005D' 38 09               jr      c,exit          ; nein
```

```
0061' 36 00                ld     (nl),0          ; reset Stunden
0061' 2A 0000#             ld     hl,(@date)
0064' 23                   inc    hl
0065' 22 0000#             ld     (@date),hl      ; Tage + 1
0068' 32 0037'     exit:   ld     (usrflg),a      ; freigeben zur Ausgabe
006B' E1           exit1:  pop    hl
006C' F1                   pop    af
006D' FB                   ei
006E' ED 4D                reti

0 Error(s) Detected. 112 Program Bytes.
199 Symbols Detected.
```

TIME.XRF

Eine weitere Besonderheit unter CP/M 3 ist es, daß eine RAM-Floppy sehr einfach angepaßt werden kann.

Auch hierzu sei das nachfolgende Listing ein Beispiel.

Zu berücksichtigen bei der Anpassung ist immer die Größe des gemeinsamen (common) Speicherbereiches, die RAM-Floppy darf niemals in diesen Bereich überlaufen. Im übrigen wird auch für eine RAM-Floppy ein DPH und ein DPB benötigt ...

```
'RAM-FLOPPY mit NDR-Klein-Computer'
RDISK    INC  3 Dec 85 09:38

              A;*******************************************************************
              A;*                                                                 *
              A;*  RAM-FLOPPY fuer NDR-Klein-Computer unter CP/M 3                *
              A;*                                                                 *
              A;*  rel 1.1              850916                                    *
              A;*                                                                 *
              A;* Copyright (C)  Raoul O. Koerber                                 *
              A;*                                                                 *
              A;*******************************************************************
              A
              A       ;
              A       ; ****************
              A       ; Variable
              A       ; ****************
              A       ;
              A       ; bei der Bankzuweisung ist zu beachten, dass ein
              A       ; CP/M3 System MINDESTENS 2 Banks (00 und 01) benoetigt
              A       ; der RAM-Floppy (RF) zugewiesene Bankbereiche duerfen
              A       ; ueber GENCPM nicht nocheinmal zugewiesen werden
              A       ;
              A
              A
        FFFF  A  IF   Z80ASM
        FFFF  A  if   askit
        0004  A        .accept 'Erste Bank     (moeglich ab Bank 2) ',first$bank
              A
              A  else
              A  endif
        FFFF  A  if   askit
              A
        000F  A        .accept 'Letzte Bank    (moeglich bis Bank 15) ',last$bank
              A
              A  else
              A  endif
              A  ENDIF
              A
        000C  Artracks      equ    last$bank-first$bank+1
        0040  Adirs         defl   64                 ; DIR-Eintrage
        0400  Aseclen       defl   1024               ; Sektorgroesse
```

```
                    A       dseg                              ; Gebankt
                    A
                    A       ;
                    A       ; ****************
                    A       ; RF-XDPH
                    A       ; ****************
                    A       ;
                    A
008C" 0140"         A       dw      rf$write
008E" 0153"         A       dw      rf$read
0090" 0176"         A       dw      rf$login
0092" 00AF"         A       dw      rf$init
0094" 00 00         A       db      0,0                       ; Keine Zuweisung
0096"               Afde:   dph     0,rdpb                    ; Kein Skew
0096" 0000          B       dw      0
0098" 00 00 00 00B          db      0,0,0,0,0,0,0,0,0
00A1" FF            B       db      -1
00A2" 0020'         B       dw      rdpb
00A4" FFFE          B       dw      -2
00A6" FFFE          B       dw      -2
00A8" FFFE  FFFE B          dw      -2,-2,-2
00AE" 00            B       db      0
                    A
                    A       cseg                              ; Gemeinsamer Bereich
                    A
         FFFF       A   if  rtracks ge 4
         0080       Adirs   defl    2*dirs
         0800       Ablklen defl    2048
0020'               Ardpb:  dpb     1024,56,rtracks,blklen,dirs,0
0020' 01C0          B       dw      ??0011
0022' 04 0F         B       db      ??0012,??0013
0024' 00            B       db      ??0014
0025' 014F          B       dw      ??0015
0027' 007F          B       dw      ??0016
0029' C0 00         B       db      ??0017,??0018
002B' 0020          B       dw      ??0019
002D' 0000          B       dw      0
002F' 03 07         B       db      ??0020,??0021
                    A
                    A   else
                    A   endif
                    A
                    A       dseg                              ; gebankter Bereich
                    A
00AF"               Arf$init:
                    A
                    A       ;
                    A       ; Zur Erkennung der RAMFLOPPY wird eine Leerdatei RDISK in
                    A       ; die Directory eingeschrieben. Bei Init wird ueberprueft
                    A       ; ob diese Datei angelegt ist.
                    A       ;
                    A
00AF" 01 0004       A       ld      bc,first$bank             ; dort ist die Directory
00B2" CD 0000#      A       call    ?xmove##
00B5" 11 0000       A       ld      de,0                      ; Quelle 1. Eintrag
00B8" 21 0119"      A       ld      hl,rf$fcb1                ; Ziel
00BB" 01 0007       A       ld      bc,cmp$len                ; muesste reichen zum Vergleichen
00BE" E5            A       push    hl                        ; Zeiger retten
00BF" CD 0000#      A       call    ?move##                   ; und Transfer
00C2" E1            A       pop     hl                        ; Zeiger auf 'Inhalt' der DIR
00C3" 11 00F9"      A       ld      de,rf$fcb0                ; Zeiger auf FCB wie gewuenscht
00C6" 06 07         A       ld      b,cmp$len                 ; soviel Zeichen zum Vergleich
```

```
00C8" 1A          Arf$in1: ld     a,(de)              ; Zeichen aus FCB0
00C9" BE          A        cp     (hl)                ; gleich FCB1?
00CA" 20 05       A        jr     nz,rf$in2           ; ... wenn nicht ...
00CC" 13          A        inc    de
00CD" 23          A        inc    hl
00CE" 10 F8       A        djnz   rf$in1              ; bis alles klar
00D0" C9          A        ret                        ; ... kein INIT notwendig
                  A
00D1" 01 0400     Arf$in2: ld     bc,first$bank shl 8 + 0
00D4" CD 0000#    A        call   ?xmove##
00D7" 11 00F9"    A        ld     de,rf$fcb0          ; original FCB=Quelle
00DA" 21 0000     A        ld     hl,0                ; Ziel
00DD" 01 0020     A        ld     bc,32               ; Laenge
00E0" CD 0000#    A        call   ?move##             ; und Transfer
00E3" 06 7F       A        ld     b,dirs-1            ; soviel DIR sind leer
00E5" C5          Arf$in3: push   bc                  ; rette DIR-Zaehler
00E6" 01 0400     A        ld     bc,first$bank shl 8 + 0
00E9" CD 0000#    A        call   ?xmove##            ; XMOVE init
00EC" 11 0120"    A        ld     de,rf$fcb2          ; Dummy-FCB
00EF" 01 0020     A        ld     bc,32               ; Laenge des Dummy
00F2" CD 0000#    A        call   ?move##             ; Transfer
00F5" C1          A        pop    bc                  ; DIR-Zaehler
00F6" 10 ED       A        djnz   rf$in3
00F8" C9          A        ret
                  A
00F9" 20 52 41 4DArf$fcb0:db      20h,'RAMDISK ',20H,20H,20H,0,0,0,0
0109" 00 00 00 00A        db      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                  A
0119"   0007      Arf$fcb1:ds     7
                  A
        0007      Acmp$len equ    $-rf$fcb1
0120"             Arf$fcb2:rept   32
0120"             A        db     0E5H
0120"             A        endm
0120" E5          B        db     0E5H
0121" E5          B        db     0E5H
0122" E5          B        db     0E5H
0123" E5          B        db     0E5H
0124" E5          B        db     0E5H
0125" E5          B        db     0E5H
0126" E5          B        db     0E5H
0127" E5          B        db     0E5H
0128" E5          B        db     0E5H
0129" E5          B        db     0E5H
012A" E5          B        db     0E5H
012B" E5          B        db     0E5H
012C" E5          B        db     0E5H
012D" E5          B        db     0E5H
012E" E5          B        db     0E5H
012F" E5          B        db     0E5H
0130" E5          B        db     0E5H
0131" E5          B        db     0E5H
0132" E5          B        db     0E5H
0133" E5          B        db     0E5H
0134" E5          B        db     0E5H
0135" E5          B        db     0E5H
0136" E5          B        db     0E5H
0137" E5          B        db     0E5H
0138" E5          B        db     0E5H
0139" E5          B        db     0E5H
013A" E5          B        db     0E5H
```

```
0138" E5         B        db      0E5H
013C" E5         B        db      0E5H
013D" E5         B        db      0E5H
013E" E5         B        db      0E5H
013F" E5         B        db      0E5H
                 A
0140"            Arf$write:
                 A
0140" 3A 0000#   A        ld      a,(@trk)              ; TRACK=Zielbank
0143" C6 04      A        add     a,first$bank          ; +Offset berechnen
0145" 47         A        ld      b,a                   ; --> Ziel
0146" 3A 0000#   A        ld      a,(@dbnk)             ; Quellbank
0149" 4F         A        ld      c,a                   ; --> Quelle
014A" CD 0000#   A        call    ?xmove##              ; XMOVE initialisieren
014D" CD 0167"   A        call    r$calc                ; Berechne RAM-Adresse
0150" C3 0000#   A        jp      ?move##
                 A
0153"            Arf$read:
                 A
0153" 3A 0000#   A        ld      a,(@trk)              ; TRACK=Quellbank
0156" C6 04      A        add     a,first$bank
0158" 4F         A        ld      c,a
0159" 3A 0000#   A        ld      a,(@dbnk)             ; Zielbank
015C" 47         A        ld      b,a
015D" CD 0000#   A        call    ?xmove##
0160" CD 0167"   A        call    r$calc
0163" EB         A        ex      de,hl
0164" C3 0000#   A        jp      ?move##
                 A
0167"            Ar$calc:
                 A
                 A        ;
                 A        ; Berechnen der Quell- bzw Zielparameter
                 A        ; zurueck mit:
                 A        ; <HL> Zeiger auf RF-SEKTOR-Anfang
                 A        ; <DE> Zeiger auf DMA-Adresse
                 A        ;
                 A
0167" 3A 0000#   A        ld      a,(@sect)             ; Sektor Nummer
016A" 67         A        ld      h,a                   ; Berechnen RAM-Start
016B" 2E 00      A        ld      l,0
016D" 29         A        add     hl,hl                 ; *2
016E" 29         A        add     hl,hl                 ; *4 = Sektoranfangsadresse
016F" ED 5B 0000#A        ld      de,(@dma)             ; DMA-Adresse
0173" 01 0400    A        ld      bc,seclen             ; Sektorgroesse
                 A
0176"            Arf$login:
                 A
0176" C9         A        ret
```

Kapitel 3 Ausblicke oder was kommt danach ...

Sicherlich läßt CP/M 3 dem geübten Benutzer gegenüber noch einige Wünsche und Fragen offen. Aber gegenüber vielen anderen Betriebssystemen hat es halt immer noch den Vorteil ausgereift und damit (relativ) fehlerfrei zu sein. Es besticht durch seine doch recht einfache Anwendung und die Tatsache, daß es CP/M auch für andere Prozessoren als den 8080 bzw. Z80 gibt. (CPM86 und CPM68).

Wenn ein vom Prozessor her nicht kompatibler Computer mit einer derart großen Verteilung (in Stückzahlen incl. Nachbauten) wie der APPLE (R) von sich behauptet der verbreitetste CP/M-Computer (wohlgemerkt mit einer Z80-Zusatzkarte) zu sein, so spricht diese Tatsache doch für sich, für 8-Bit Computer gibt es nichts Besseres, zumindest nichts zu diesem Preis und mit dieser Verbreitung.

Es ist auch immer wieder verwunderlich, daß 16-Bit-Computer wie der IBM-PC (R) und der ATARI ST (R) verblüffend schnell CP/M-Emulationsprogramme zur 'Verfügung' haben.

Was die Wünsche angeht, so bietet CP/M mit der Möglichkeit des Einsatzes von RSX-Modulen sicherlich ein recht brauchbares Instrument um (fast) alles zu erzwingen, was dem Einzelnen fehlen mag.

Vielleicht am 'schwächsten' ist der CCP, hier würden gar Viele es gerne sehen, wenn er noch mehr könnte, aber auch hier bleibt die Frage offen, wer hätte davon den Nutzen? Der normale Benutzer würde sicher durch mehr Möglichkeiten auch mehr verwirrt werden und ist es nicht die vornehmste Ausgabe eines Betriebssysteme Programme einwandfrei ablaufen zu lassen die für das Betriebssystem geschrieben wurden, anstelle mit noch mehr Optionen (in der CCP-Befehlszeile) irgendwelche Daten mit irgendwelchen Zusatzfunktionen durch irgendwelche Zusatzkanäle zu schleusen?

Ganz sicher stellt sich CP/M 3 als ein recht zuverlässiges System dar, mit dem der Anfänger mit wenig Einarbeitung und der Profi ohne Plage gut zurecht kommt - und als mehr hat sich CP/M auch nie dargestellt.

Wer bereits mit einem CP/M3-System gearbeitet hat, daß auf eine RAM/ROM-Floppy bootet und Disketten nur noch als Langzeitspeicher benutzt, der kann sich sicherlich nur sehr schwer an die langsamen Bearbeitungszeiten mancher 16-Bit-Rechner gewöhnen.

Teil VI Nützliches

Kapitel 1 Die ELMON-Anpassung MONIO

Es ist im Prinzip einerlei wie der Ein- Ausgabeteil eines Monitorprogrammes aufgebaut ist. Da alle bisherigen Programmsegmente auf den NDR-Klein-Computer zugeschnitten wurden, so soll das entsprechende Listing des CP/M-3 hardware-Teibers aus dem Monitor-Programm ELMON ergänzend abgedruckt werden.

Ähnlich wie beim BDOS ist hier ebenfalls eine Sprungleiste zu erkennen, über welche alle infrage kommenden Unterprogramme aufgerufen werden können. Auch die Aufruf-Konventionen wurden weitgehend beibehalten.

Sprungleisten haben sich für derartige Anwendungen besonders gut bewährt, da die Bezugsadresse für den Aufruf immer gleich bleibt, die Unterprogramme sich jedoch mit jeder Revision, die vorgenommen wird, ändern können, ohne daß irgendwelche Anpassungen vorgenommen werden müssen.

Das nachfolgende Programmsegment ist die (adressenmäßig festgelegte) Hardware-Anpassung für den CP/M 3 Bootmonitor ELMON aus dem ELEKTRONIKLADEN, Detmold, einem der authorisierten Händler des NDR-Klein-Computers.

```
                                INCLUDE MONEQU.LIB
                        A       .list

                        ; ****************************************************************
                        ; * hardware-Ein/Ausgabe Modul fuer NDR-Projekt                  *
                        ; ****************************************************************

                                ;
                                ; Dies ist ein 'eigenstaendiger' Ein/Ausgabetreiber
                                ; incl der zugehoerigen Initialisierung.
                                ; Wird ELMON als 'Aufsatz' zu FLOMON betrieben, werden
                                ; Initialisierung und I/O soweit moeglich von dort
                                ; Dieser Treiberteil residiert ab Adresse IOSEG
                                ; und wird von CP/M3 (in der ELEKTRONIKLADEN-Version) benutzt.
                                ;

          FC00                  org     0fc00h              ; Start MONIO

                                ;
                                ; ****************
                                ; * Einspruenge  *
                                ; ****************
                                ;

                                ;
                                ; Einige EIN/AUS-Gabe Routinen werden der Kompatibilitaet halber
                                ; auf aehnliche Art wie bei FLOMON aus dem ROM-Bereich des FLOMON
                                ; 'geholt' - Bei spaeteren Aenderungen koennen die Einsprungadresse
                                ; u.U. verschoben werden - Ueber die Funktion der angesprungenen
                                ; Unterprogramme kann hier keine Aussage gemacht werden ......
                                ;

          00DE          fci     equ     00deh           ; im FLOMON CONIN
          0321          fco     equ     0321h           ; im FLOMON CONOUT
          00BD          fctst   equ     00bdh           ; im FLOMON CONIST
          0205          fri     equ     0205h           ; im FLOMON
          020E          fpo     equ     020eh           ; im FLOMON
          0218          flo     equ     0218h           ; im FLOMON


                                ;
                                ; **** CP/M kompatible Ein/Ausgaberoutinen ****
                                ; Eingabe in <A>kku Ausgabe ueber <C> ( Zeichen zurueck in <A>
                                ; Status 00=es liegt nichts vor - FF= Zeichen liegt an
                                ;
                                ; Die Reihenfolge der Sprungtabelle darf keinesfalls
                                ; veraendert werden, da alle anderen Programmteile fuer
                                ; I/O nur darauf zugreifen.
                                ;

   FC00  C3 FC3F        conist: jp      .conist         ; Konsolenstatus (EIN)
   FC03  C3 FC4E        conin:  jp      .conin          ; Konsoleneingabe
   FC06  C3 FC5D        conout: jp      .conout         ; Konsolenausgabe
   FC09  C3 FC91        list:   jp      .list           ; Druckerausgabe (CENT)
   FC0C  C3 FC87        auxout: jp      .auxout         ; Aux Ausgabe
   FC0F  C3 FC7F        auxin:  jp      .auxin          ; Aux Eingabe
   FC12  C3 FC77        auxist: jp      .auxist         ; Aux(eingabe)Status
   FC15  C3 FCA0        lstst:  jp      .lstst          ; Druckerstatus (SER)
```

```
FC18  C3 FC74     conost: jp      .conost         ; Konsolenstatus (AUS)
FC1B  C3 FCA6     casist: jp      .casist         ; Kassetten(eingabe)Status
FC1E  C3 FCAB     casin:  jp      .casin          ; Kasseteneingabe
FC21  C3 FCB3     casout: jp      .casout         ; kassettenausgabe
FC24  C3 FCBD     setbnk: jp      .setbnk         ; Bank setzen
FC27  C3 FCDC     floppy: jp      .floppy         ; Disk-Treiber

                          ;
                          ; Reserviert fuer spaetere Erweiterungen
                          ;

FC2A  C3 0000             jp      $-$
FC2D  C3 0000             jp      $-$
FC30  C3 FCD3     init:   jp      .init

                          ;
                          ; Flags und Zeiger
                          ;

FC33  00          iobyte: db      0               ; z.Z. nicht implementiert
FC34  00          curbnk: db      0               ; gueltige Bank
FC35  00          @dbnk:  db      0
FC36  00          @cbnk:  db      0

FC37  00          flg0:   db      0               ; Reserve
FC38  00          flg1:   db      0
FC39  00          flg2:   db      0
FC3A  00          flg3:   db      0
FC3B  00          flg4:   db      0
FC3C  00          flg5:   db      0
FC3D  00          flg6:   db      0
FC3E  00          flg7:   db      0

                          ;
                          ; ********************************
                          ; * Ein/Ausgabe Routinen         *
                          ; ********************************
                          ;



                          ;
                          ; ****************
                          ; * Konsolstatus *
                          ; ****************
                          ;

FC3F              .conist:

                          ;
                          ; diese Routine bringt FF zurueck wenn ein Zeichen Eingabebereit
                          ; ist, andernfalls kommt 00 zurueck
                          ; Abgefragt wird KEY
                          ;

FC3F  ED 73 FE7E          ld      (spsav),sp      ; zuerst Stack nach 'oben'
FC43  31 FEC0             ld      sp,tstack
FC46  CD FCC2             call    open            ; dann ROM oeffen
FC49  CD 00BD             call    FCTST
```

```
FC4C  18 1C                 jr      cc
                            ;
                            ; ***************
                            ; *Konsoleingabe *
                            ; ***************
                            ;
FC4E              .conin:
                            ;
                            ; Eingaberoutine wartet bis Zeichen anliegt
                            ; Bringt Zeichen in <A> zurueck - ohne Echo auf Bildschirm
                            ;
FC4E  ED 73 FE7E            ld      (spsav),sp
FC52  31 FEC0               ld      sp,tstack
FC55  CD FCC2               call    open
FC58  CD 000E               call    FCI
FC5B  18 0D                 jr      cc
FC5D              .conout:
                            ;
                            ; Zeichenausgabe ueber <C>
                            ;
FC5D  ED 73 FE7E            ld      (spsav),sp      ; rette Benutzer Stack
FC61  31 FEC0               ld      sp,tstack       ; .. denn drinnen wird's .....
FC64  CD FCC2               call    open
FC67  CD 0321               call    FCO
                            ;
                            ; <A> retten und ROM 'zu'
                            ;
FC6A  F5          cc:       push    af
FC6B  CD FCC9               call    close           ; schliesse ROM-Bereich
FC6E  F1                    pop     af
FC6F  ED 7B FE7E            ld      sp,(spsav)      ; Benutzer Stack
FC73  C9                    ret
FC74              .conost:
                            ;
                            ; GDP ist 'immer' bereit
                            ;
FC74  AF                    xor     a
FC75  3D                    dec     a               ; FF ohne ZERO-Flag
FC76  C9                    ret
                            ;
                            ; ***************
                            ; * AUX Kanal    *
                            ; ***************
                            ;
```

```
                              ; AUX als SER-ielle Schnittstelle
                              ;

FC77                   .auxist:

                              ;
                              ; AUX Eingabe-Status
                              ;

FC77  DB ED                   in      a,(ustal)
FC79  E6 08                   and     00001000b
FC7B  C8                      ret     z               ; nicht bereit
FC7C  F6 FF                   or      -1
FC7E  C9                      ret

FC7F                   .auxin:

                              ;
                              ; AUX Eingabe
                              ;

FC7F  CD FC77                 call    .auxist
FC82  28 FB                   jr      z,.auxin
FC84  DB EC                   in      a,(udatl)
FC86  C9                      ret                     ; ohne Parity-Maskierung

FC87                   .auxout:

                              ;
                              ; AUX Ausgabe
                              ;

FC87  DB ED                   in      a,(ustal)
FC89  E6 10                   and     00010000b
FC8B  28 FA                   jr      z,.auxout       ; ... noch nicht bereit
FC8D  79                      ld      a,c             ; Zeichen uebernehmen
FC8E  D3 EC                   out     (udatl),a       ; ausgeben
FC90  C9                      ret

                              ;
                              ; ***************
                              ; * Drucker     *
                              ; ***************
                              ;

FC91                   .list:

                              ;
                              ; Druckerausgabe
                              ;

FC91  ED 73 FE7E              ld      (spsav),sp
FC95  31 FEC0                 ld      sp,tstack
FC98  CD FCC2                 call    open
FC9B  CD 0218                 call    FLO
FC9E  18 CA                   jr      cc

```

```
FCA0                  .lstst:


                                ;
                                ; Drucker Status
                                ;

FCA0  DB 49                     in      a,(censtb)
FCA2  0F                        rrca                        ; wenn CARRY=nicht bereit
FCA3  3F                        ccf                         ; daher wechseln
FCA4  9F                        sbc     a,a                 ; X-X=0 -C=FF !!!
FCA5  C9                        ret

                                ;
                                ; ****************
                                ; * kassette     *
                                ; ****************
                                ;

FCA6                  .casist:

                                ;
                                ; kassetten Eingabe-Status
                                ;

FCA6  DB CA                     in      a,(cmdcas)          ; Status lesen
FCA8  0F                        rrca                        ; RX in Carry
FCA9  9F                        sbc     a,a
FCAA  C9                        ret

FCAB                  .casin:

                                ;
                                ; Kassetteneingabe Zeichen in (A)
                                ;

FCAB  CD FCA6                   call    .casist             ; rx-Status pruefen
FCAE  28 FB                     jr      z,.casin            ; warten bis bereit
FCB0  DB CB                     in      a,(datcas)
FCB2  C9                        ret

FCB3                  .casout:

                                ;
                                ; kassettenausgabe
                                ;

FCB3  DB CA                     in      a,(cmdcas)          ; tx-Status pruefen
FCB5  0F                        rrca                        ; tx in Carry
FCB6  0F                        rrca
FCB7  30 FA                     jr      nc,.casout          ; warten bis frei
FCB9  79                        ld      a,c                 ; Zeichenuebergabe
FCBA  D3 CB                     out     (datcas),a          ; Ausgabe
FCBC  C9                        ret

                                ;
                                ; ****************
                                ; * Bootkarte    *
                                ; ****************
```

```
                                ;


                                ; Die BOOT-Karte enthaelt FLOMON und ELMON darueberhinaus
                                ; einen derzeit nur von FLOMON benutzten RAM-Bereich
                                ; Beide Programme UND der RAM-Bereich (wegen GDP) sind zum
                                ; Betrieb von CP/M3 notwendig.
                                ; Die BOOT-Karte uebernimmt darueberhinaus noch die Bankumschaltung
                                ;

FCBD                    .setbnk:

                                ;
                                ; Bank wird entsprechend dem Inhalt in (A) gesetzt
                                ; Diese gueltige Bankadresse wird in der Variablen
                                ; CURBNK abgelegt um bei CLOSE entsprechend schalten
                                ; zu koennen. Gleichzeitig wird EPROM abgeschaltet
                                ;

FCBD  32 FC34                   ld      (curbnk),a      ; Ablegen
FCC0  18 0A                     jr      cl1

FCC2                    open:

                                ;
                                ; Booteprom freigeben zum Zugriff
                                ;

FCC2  3A FC34                   ld      a,(curbnk)      ; derzeitige Bank
FCC5  CB BF                     res     7,a             ; oeffnen
FCC7  18 05                     jr      cl2

FCC9                    close:

                                ;
                                ; Boot-Eprom abschalten
                                ;

FCC9  3A FC34                   ld      a,(curbnk)      ; derzeitige Bank
FCCC  CB FF             cl1:    set     7,a             ; schliessen
FCCE  D3 C8             cl2:    out     (bbport),a
FCD0  CB BF                     res     7,a             ; Original zurueckgeben
FCD2  C9                        ret

                                ;
                                ; ***************
                                ; * INIT        *
                                ; ***************
                                ;

FCD3                    .init:

                                ;
                                ; INIT wird bereits von FLOMON uebernommen
                                ; es bleibt hier nur AUX zu initialisieren
                                ;

FCD3  3E 9E                     ld      a,10011110b     ;2Stop-8Bit-int GEN-9600baud
```

```
FCD5  D3 EF          out    (uconi),a
FCD7  3E 0B          ld     a,00001011b      ;keine Parity-normal-disINT-DTRlow
FCD9  D3 EE          out    (ucmol),a
FCDB  C9             ret
```

```
                      ; ***********************************************************
                      ; *                                                         *
                      ; * Dies ist ein eigenstaendiger Floppytreiber der NICHT im *
                      ; * Interrupt-Betrieb arbeitet und eine Bankumschaltung (wie in *
                      ; * CPM3 verlangt) vornimmt.                                *
                      ; *                                                         *
                      ; * Es wurde bewusst der Syntax des FLOMON beibehalten um   *
                      ; * von dieser Seite her kompatibel zu bleiben.             *
                      ; *                                                         *
                      ; * Als Einschraenkung zur kompatibilitaet gilt lediglich die *
                      ; * neue Einsprungadresse.                                  *
                      ; *                                                         *
                      ; ***********************************************************

                              .comment %

                      Programmteile in GROSSCHRIFT sind ORGINAL-Code aus dem
                      FLOMON1 (V1.6) von Rolf-Dieter Klein.
                      Einer moeglichst weitgehenden Kompatibilitaet wegen wurde
                      soweit moeglich der Original-Code uebernommen. Wesentliche
                      Aenderungen wurden im physikalischen Diskkettenzugriff
                      vorgenommen da unter CPM3 nicht im Interrupt MODE 1 gearbeitet
                      werden kann  -und MODE 2 nicht moeglich ist. Alle Treiber-
                      Routinen wurden fuer POLLING geschrieben. Darueberhinaus wurde
                      die Bankumschaltung beruecksichtigt.

                      Einige Aenderungen wurden darueberhinaus vorgenommen um den
                      das Programm kompakter zu bekommen - oder auch der Uebersichtlichkeit
                      halber. Diese Programmteile sind in Kleinschrift

                      850801 Raoul O. Koerber nach Absprache mit Herrn Klein

                      FDCSEL   als Leseport

                      7       6       5       4       3       2       1       0

                      DRQ   INTRQ   HEADLD   unbelegt
                      0=request
                              0=request
                                      1=head down

                      FDCSEL   als Schreibport

                      7       6       5       4       3       2       1       0

                      sso     moton   mini    dense   d       c       b       a

                      1=Seite 1(2)
                              0=Motor an
                                      1=Minilaufwerk
                                              1=Single Dense
                              %

          FFFF        neu     equ     0ffffh

                              ;
                              ; ***************
                              ; * Floppy      *
```

```
                                 ; ****************
                                 ;


     FCDC              FLOPPY:

                                 ;
                                 ; das wesentliche an diesem Treiber ist, dass er OHNE Interrupts
                                 ; arbeitet --- der Kontroller 179x gibt jedoch nach Befehlsab-
                                 ; arbeitung IMMER einen Interrupt aus ... dies kann bei der
                                 ; Kontrollerkarte aus dem ELEKTRONIKLADEN mechanisch (Jumper) unter-
                                 ; drueckt werden, bei FL02 wird dies jedoch etwas kompliziert -
                                 ; es wurde daher eine software-Loesung vorgezogen um IMMER
                                 ; kompatibel zum Booten ueber FLOMON fuer CPM2 zu bleiben
                                 ;

     FCDC  F3                    di                        ; kein Interrupt
     FCDD  3E D0                 ld      a,forcei          ; Reset Kontroller
     FCDF  D3 C0                 out     (fdccmd),a
     FCE1  ED 56                 im      1                 ; im normalen I-modus bleiben
     FCE3  CD FD8C               call    softex            ; jetzt Befehl ausfuehren
     FCE6  F5                    push    af                ; rette Fehlerstatus
     FCE7  DB C4       flop1:    in      a,(fdcsel)        ; liegt ein INT an?
     FCE9  CB 77                 bit     6,a
     FCEB  28 04                 jr      z,flop2           ; ... wenn nicht
     FCED  DB C0                 in      a,(fdccmd)        ; sonst Interrupt ruecksetzen
     FCEF  18 F6                 jr      flop1

     FCF1  F1          flop2:    pop     af                ; Fehlerstatus
     FCF2  FB                    ei                        ; nun interrupts wieder zulassen
     FCF3  C9                    ret


     FCF4              SETUP:

                                 ;
                                 ; Sektor Register Laden - und Laufwerk ansprechen
                                 ;

     FCF4  3A FC35               ld      a,(@dbnk)         ; Zielbank
     FCF7  CD FC24               call    setbnk            ; setzen
     FCFA  CD FE41               call    dsel              ; nun Laufwerk ansprechen
     FCFD  F5                    push    af                ; Rette SELECT
     FCFE  7B                    ld      a,e               ; Sektorregister setzen
     FCFF  D3 C2                 out     (fdcsec),a
     FD01  0E C3                 ld      c,fdcdat          ; Adressport nach <C>

                                 ;
                                 ; testet ob der Kopf noch  geladen ist und setzt die Seite
                                 ; Entscheidung fuer sso-Bit ist sideselect in Register <C>
                                 ;

     FD03  06 00                 LD      B,00000000b       ; Flag fuer SSO-Bit
     FD05  F1                    pop     af                ; SELECT
     FD06  CB 7F                 bit     7,a               ; teste side-select
     FD08  28 02                 JR      Z,SETIUP          ; wenn Seite 0(1)
     FD0A  06 02                 LD      B,00000010b       ;    muss sonst Seite 1(2) sein ..

```

```
FD0C                 SETTUP:
                                ;
                                ; pruefen ob kopf 'unten' sonst zusaetzlich DELAY
                                ;
FD0C  DB C4                     IN     A,(FDCSEL)        ; Status Port
FD0E  E6 20                     AND    00100000b         ; Head-Load Bit
FD10  C0                        ret    nz                ; ... wenn bereits auf Diskette
FD11  CB D0                     set    2,b               ; sonst +15ms delay ...
FD13  C9                        RET
FD14                 RDFLP:
                                ;
                                ; Floppy-Sektor lesen
                                ;
FD14  CD FCF4                   CALL   SETUP             ; Register und sso vorbereiten
FD17  18 1F                     jr     getdata           ; Poll nach Daten
FD19                 WRFLP:
                                ;
                                ; Floppy-Sektor schreiben
                                ;
FD19  CD FCF4                   CALL   SETUP             ; Register und sso vorbereiten
FD1C  3E A8                     LD     A,writes          ; write
FD1E  B0                        OR     B                 ; sso
FD1F  D3 C0                     OUT    (FDCCMD),A        ; Schreibbefehl ausgeben
FD21  DB C4          WR1:       IN     A,(FDCSEL)        ; warten bis DRQ da
FD23  07                        RLCA
FD24  30 FB                     JR     NC,WR1            ; dann Wert ausgeben
FD26  DB C0          wr2:       in     a,(fdccmd)        ; nun POLL auf DRQ
FD28  CB 4F                     bit    1,a
FD2A  28 05                     jr     z,wr3             ; POLL auf BUSY
FD2C  ED A3                     OUTI
FD2E  C3 FD26                   jp     wr2               ; ... weiter
FD31  CB 47          wr3:       bit    0,a               ; evtl am Ende?
FD33  C2 FD26                   jp     nz,wr2            ; wenn nicht ...
FD36  18 1A                     jr     rwdone            ; sonst gemeinsamer Aussprung
FD38                 getdata:
FD38  3E 88                     LD     A,reads
FD3A  B0                        OR     B
FD3B  D3 C0                     OUT    (FDCCMD),A
FD3D  DB C4          RD1:       IN     A,(FDCSEL)        ; warten bis DRQ
FD3F  07                        RLCA
FD40  30 FB                     JR     NC,RD1
FD42  DB C0          rd2:       IN     A,(FDCCMD)        ; Status
FD44  CB 4F                     bit    1,a               ; DRQ?
FD46  28 05                     jr     z,rd3             ; wenn nicht ..
FD48  ED A2                     INI
```

```
FD4A  C3 FD42              jp      rd2             ; naechstes Byte

FD4D  CB 47        rd3:    bit     0,a             ; noch BUSY?
FD4F  C2 FD42              jp      nz,rd2          ; ... wenn ja

FD52  F5           rwdone: push    af              ; Rette Fehlercode
FD53  3A FC36              ld      a,(@cbnk)       ; CPU-Bank
FD56  CD FC24              call    setbnk          ; wieder einstellen
FD59  F1                   pop     af              ; Fehlercode zurueck
FD5A  E6 FC                and     11111100b       ; setze Fehlerflags
FD5C  C9                   ret

FD5D               DELAY:

                           ;
                           ; Warteschleife je Durchgang ca 1ms (bei 4MHz)
                           ;

FD5D  C5                   push    bc
FD5E  06 E6                ld      b,230
FD60  00           del1:   nop
FD61  10 FD                djnz    del1
FD63  C1                   pop     bc
FD64  0B                   DEC     BC
FD65  78                   LD      A,B
FD66  B1                   OR      C
FD67  20 F4                jr      NZ,DELAY
FD69  C9                   RET

FD6A               SEEKEX:

                           ;
                           ; Spur anfahren C=Drive D=Spur
                           ;

FD6A  C5                   push    bc
FD6B  7A                   LD      A,D             ; Track
FD6C  D3 C3                OUT     (FDCDAT),A      ; ins Datenregister fuer ...
FD6E  06 1C                LD      B,seek          ; ...Seek-Befehl
FD70  CD FD77              call    doit
FD73  C1                   pop     bc
FD74  C9                   ret

FD75               RSTORE:

                           ;
                           ; Spur 0 anfahren
                           ;

FD75  06 08                ld      b,home


FD77               doit:
                           ;
                           ; Ausfuehrung RESTORE und SEEK
                           ;

FD77  CD FE41              call    dsel            ; Laufwerk ansprechen
```

```
FD7A  3A FE6A             ld     a,(steprate)    ; Steppingrate
FD7D  B0                  or     b               ; in FDC-Befehl integrieren
FD7E  D3 C0               out    (fdccmd),a      ; ... ausfuehren

FD80  DB C0     busy:     in     a,(fdccmd)      ; ... erst warten bis BUSY
FD82  0F                  rrca
FD83  30 FB               jr     nc,busy
FD85  DB C0     busy1:    in     a,(fdccmd)      ; und dann warten bis fertig
FD87  CB 47               bit    0,a
FD89  20 FA               jr     nz,busy1
FD8B  C9                  ret


FD8C            SOFTEX:


                          ;
                          ; Floppy-Kernroutinen
                          ; hl=adresse, d=track e=sektor
                          ; b=Befehl 0=Steprate setzen 1=read 2=write
                          ; c=Laufwerkscode
                          ;

FD8C  78                  LD     A,B             ; Befehl
FD8D  B7                  OR     A               ; ... setze steppingrate ?
FD8E  20 08               JR     NZ, SOFT1       ; ... wenn nicht

                          ;
                          ; im Original-Treiber ist Bit 7 das sso-Flag - wird aber
                          ; nirgends ernsthaft benoetigt, da IMMER verlangt wird, dass
                          ; side-select im SELECT-CODE (Register <C>) gesetzt ist.
                          ; In diesem Treiber muss Funktion SET-STEPPING-RATE (B=0)
                          ; nur Erstmals (falls anders als 3ms) oder bei Aenderung
                          ; aufgerufen werden.
                          ;

FD90  7A                  LD     A,D             ; Uebergabe in <D>
FD91  E6 03               AND    00000011B       ; sicherheitshalber wegen 'altem' sso
FD93  32 FE6A             LD     (STEPRATE),A    ; stepping-rate ablegen
FD96  AF                  XOR    A               ; Fehlerfrei ...
FD97  C9                  RET                    ; zurueckmelden

FD98            SOFT1:


                          ;
                          ; Schreiben oder Lesen
                          ; es sei nachfolgend unterschieden zwischen:
                          ;      SELECT          = Laufwerkcode-Dichte-Laufwerkstyp-sideselect
                          ;      Laufwerkscode   = Bit 0-3, entscheidet welches Laufwerk an-
                          ;                        gesprochen wird
                          ;

FD98  AF                  XOR    A
FD99  32 FE68             LD     (FEHZA),A       ; Fehlerzaehler zuruecksetzen
FD9C  79                  LD     A,C             ; SELECT
FD9D  32 FE6C             LD     (CCODE),A       ; ablegen
FDA0  E6 0F               AND    00001111b       ; fuer Vergleich nur Laufwerkscode
FDA2  4F                  LD     C,A             ; ... zurueck nach <C>
FDA3  3A FE6D             LD     A,(DRVAT)       ; ist das angesprochene Laufwerk
```

```
FDA6  FE FF                 CP     neu          ; ... noch 'unbenutzt' ?
FDA8  28 13                 JR     Z,NOTDEF     ;     dann einloggen ..
FDAA  E6 0F                 AND    00001111b    ; nur Laufwerkscode relevant
FDAC  B9                    CP     C            ; Vergleich ob Laufwerkewechsel
FDAD  28 34                 JR     Z,NURSEEK    ;   wenn gleich - nur SEEK notwendig
FDAF  E5                    PUSH   HL           ; rette DMA-Adresse
FDB0  C5                    PUSH   BC           ;       Befehl und Laufwerkscode
FDB1  21 FE6E               LD     HL,DRVTAB    ; Basisadresse der Laufwerkstabelle

                            ;
                            ; in Laufwerkstabelle wird der momentane Inhalt des
                            ; Trackregisters abgelegt
                            ;

FDB4  4F                    LD     C,A
FDB5  06 00                 LD     B,0
FDB7  09                    ADD    HL,BC        ; Berechne Adresse in Tabelle
FDB8  DB C1                 IN     A,(FDCTRK)   ; lese Track aus Trackregister
FDBA  77                    LD     (HL),A       ; und lege in Tabelle ab
FDBB  C1                    POP    BC
FDBC  E1                    POP    HL

FDBD               NOTDEF:

                            ;
                            ; es wird hier nachgeprueft ob das Laufwerk 'vorhanden' ist
                            ; Das Trackregister wird mit dem Inhalt der Laufwerkstabelle
                            ; geladen. Beim ersten Laden ist dies ein FF - also ein 'nicht'
                            ; moeglicher Track - dies wird bei SEEK jedoch erkannt ...
                            ;

FDBD  E5                    PUSH   HL
FDBE  C5                    PUSH   BC
FDBF  21 FE6E               LD     HL,DRVTAB    ; Tabellenstart
FDC2  06 00                 ld     b,0          ; (C) ist Laufwerkscode
FDC4  09                    ADD    HL,BC
FDC5  7E                    LD     A,(HL)       ; lese 'letzten' Track
FDC6  D3 C1                 OUT    (FDCTRK),A   ; diesen ins Trackregister des FDC
FDC8  C1                    POP    BC
FDC9  E1                    POP    HL
FDCA  3A FE6C               LD     A,(CCODE)    ; das angesprochene Laufwerk
FDCD  32 FE6D               LD     (DRVAT),A    ; ist nun das 'neue' Laufwerk

FDD0  C5                    PUSH   BC
FDD1  01 0001               ld     bc,1         ; 1 ms warten
FDD4  CD FD5D               CALL   DELAY
FDD7  CD FE41               call   dsel         ; nun Laufwerk ansprechen

                            ;
                            ; nachfolgender Delay hat an dieser Stelle wenig Sinn, wenn
                            ; der Motor nicht mit DSEL anlaeuft .... ist bei entsprechender
                            ; Jumper-Stellung bei der FDC-Karte aus dem ELEKTRONIKLADEN
                            ; jedoch der Fall ...
                            ;

FDDA  01 0023               LD     BC,35        ; ca 35ms .. 15ms
FDDD  CD FD5D               CALL   DELAY        ; um head-load-Zeit zu beruecksichtigen
FDE0  C1                    POP    BC           ; Befehl und Laufwerkscode
```

```
FDE1  18 06             JR      NUR1            ; teste auf READY


FDE3                NURSEEK:

                        ;
                        ; Laufwerk und Track sind erkannt - nun pruefen ob
                        ; Kopf aufliegt
                        ;

FDE3  DB C4             IN      A,(FDCSEL)      ; wenn Kopf unten
FDE5  E6 20             AND     00100000b
FDE7  20 06             JR      NZ,NODUM        ; ist kein Dummy-SEEK notwendig

FDE9                NUR1:

                        ;
                        ; hier wird ein SEEK veranlasst - denn nur so kann das
                        ; Vorhandensein einer Diskette bei FL02 geprueft werden,
                        ; da die Laufwerksansprache NUR in Verbindung mit SELECT
                        ; und HEADLOAD erfolgt ... bei 'hard'-verdrahtetem READY
                        ; ist dies jedoch keine Loesung ....
                        ;

FDE9  CD FD6A           CALL    SEEKEX          ; SEEK um Laufwerk auf READY zu testen
FDEC  07                rica                    ; READY?
FDED  38 FA             jr      c,nur1          ; .... nochmal

                        ;
                        ; an dieser Stelle ist im Originalprogramm eine endlose
                        ; Schleife - ein 'versehentliches' Ansprechen eines Lauf-
                        ; werkes das es nicht 'gibt' oder in dem keine Diskette
                        ; eingelegt ist, fuehrt unweigerlich zum Programmabsturz
                        ; Ein TIME-OUT kann leider nur mit hardware-Aenderungen
                        ; eingefuehrt werden, da eine Ueberpruefung ob eine Diskette
                        ; eingelegt ist NUR bei laufendem Motor erfolgen kann ....
                        ;

FDEF                NODUM:

                        ;
                        ; Hierher wenn Kopf unten und Laufwerk READY
                        ; Es wird geprueft ob ein echter SEEK ausgefuehrt werden
                        ; muss oder evtl zuerst ein RESTORE.
                        ;

FDEF  DB C1             IN      A,(FDCTRK)      ; was steht im Trackregister
FDF1  FE FF             CP      neu             ; nicht definiert - neues Laufwerk
FDF3  28 0A             JR      Z,TRY1          ; ... dann zuerst RESTORE
FDF5  BA                CP      D               ; oder schon auf dem Track
FDF6  28 17             JR      Z,SK11          ; ja, dann lesen oder schreiben

FDF8                TRY:

                        ;
                        ; SEEK ausfuehren
                        ;

```

```
FDF8  CD FD6A            CALL   SEEKEX          ; SEEK
FDFB  E6 90              AND    10010000B       ; READY und RNF(SEEK)-Fehler
FDFD  28 10              JR     Z,SK11          ; ja dann R/W moeglich

FDFF              TRY1:


                         ;
                         ; Laufwerk normieren und RSTORE
                         ;

FDFF  C5                 PUSH   BC
FE00  CD FD75            CALL   RSTORE
FE03  C1                 POP    BC
FE04  CD FE39            call   seterr          ; Fehlerzaehler +1
FE07  FE 0A              CP     10              ; bei 10 Fatal-Fehler
FE09  38 ED              JR     C,TRY           ; sonst nochmals mit SEEK probieren


                         ;
                         ; SEEK-Fehler werden als hardware-Fehler mit Code FF,
                         ; gesetztem CARRY-Flag und NZ-Flag zurueckgemeldet
                         ;

FE0B  AF          nogood: xor   a
FE0C  3D                 dec    a               ; Fehlerflag bei SEEK oder RESTORE
FE0D  37                 SCF                    ; 0ffh, carry als Fehlerflag
FE0E  C9                 RET                    ; ABBRUCH FATAL

FE0F              SK11:


                         ;
                         ; Laufwerk und Track erkannt numb r/w ausfuehren
                         ;

FE0F  3A FE6C            ld     a,(ccode)       ; SELECT
FE12  4F                 ld     c,a             ; nach (C)
FE13  E5                 push   hl
FE14  C5                 push   bc
FE15  05                 dec    b               ; wird 0 wenn B=1 (read)
FE16  20 07              jr     nz,WRTEX        ; Schreibbefehl


                         ;
                         ; lesen eines physikalischen Sektors
                         ;

FE18  CD FD14            CALL   RDFLP           ; und Lesen
FE1B  E6 9C              AND    10011100B       ; Fehlermaske ohne Fehler 6&7 (wp-wf)
FE1D  18 03              jr     reterr

FE1F              WRTEX:


                         ;
                         ; Schreiben eines physikalischen Sektors
                         ;

FE1F  CD FD19            CALL   WRFLP
FE22  C1          reterr: POP   BC
FE23  E1                 POP    HL
FE24  C8                 ret    z               ; wenn Fehlerfrei ...
```

```
FE25  32 FE69           ld      (errcod),a      ; sonst Fehlercode ablegen
FE28  CB 67             bit     4,a             ; RNF-Fehler ?
FE2A  20 D3             JR      NZ,TRY1         ; dann gleich neuen Seek ausfuehren
FE2C  CD FE39           call    seterr          ; andere Fehler zaehlen
FE2F  FE 0B             CP      11              ; 10 Versuche zulassen
FE31  38 DC             jr      c,sk11          ;    ... Fehlerschleife (10*)
FE33  3A FE69           ld      a,(errcod)      ; letzen (Schreib-Lesefehler)
FE36  B7                or      a
FE37  37                scf                     ; Fehlerflags
FE38  C9                ret

FE39            seterr:

                        ;
                        ; update des Fehlerzaehlers
                        ;

FE39  3A FE6B           ld      a,(fehza)
FE3C  3C                inc     a
FE3D  32 FE6B           ld      (fehza),a
FE40  C9                ret

FE41            dsel:

                        ;
                        ; Ansprechen des Laufwerkes. SELECT wird in CCODE erwartet
                        ;

FE41  3A FE6C           ld      a,(ccode)
FE44  CB F7             set     6,a             ; Reset MOTOR-ON Bit
FE46  D3 C4             out     (fdcsel),a      ; wegen MV in FDC aus dem ELEKTRONIKLADEN
FE48  CB B7             res     6,a             ; denn dieser muss getoggelt werden
FE4A  D3 C4             out     (fdcsel),a      ; hat aber keinen Einfluss auf FLO2
FE4C  C9                ret

                        ;
                        ; ****************
                        ; * BOOT         *
                        ; ****************
                        ;

FE4D            boot:

                        ;
                        ; liest ersten Sektor auf einer MINI-Diskette
                        ; fuer MAXI unter CPM3 nicht vorgesehen da nur single-density
                        ; SICHER moeglich .... wir verwenden den ELMON-Stack
                        ; jedoch nur bis zum Einsprung in CPMLDR ...
                        ;

FE4D  AF                xor     a               ; Sicher Bank 0 einstellen
FE4E  CD FC24           call    setbnk
FE51  16 00             ld      d,0             ; steppingrate 3(6)ms auf Seite 0
FE53  06 00             ld      b,0             ; Befehl 0 ( set step )
FE55  CD FCDC           call    .floppy         ; direkt ...
FE58  21 F000           ld      hl,secbuf
FE5B  E5                push    hl              ; ist auch EXECUTE-Adresse
FE5C  01 0121           ld      bc,121h         ; Befehl=Lesen Laufwerk 'A' Seite 0 MINI
```

```
FE5F  11 0001          ld      de,0001h       ; Track 0 Sektor 1
FE62  CD FCDC          call    .floppy
FE65  D0               ret     nc             ; wenn ohne Fehler --- ausfuehren
FE66  C3 0000          jp      entry          ; sonst BOOT-Fehler ( mit ??? gemeldet )

                       ;
                       ; ***************
                       ; * RAM         *
                       ; ***************
                       ;

FE69  00          errcod: db      0
FE6A  00          STEPRATE:DB     0      ; Merker fuer die Steprate + SSO-Option
FE6B  00          FEHZA:  DB      0      ; Fehlerzaehler fuer retry
FE6C  00          CCODE:  DB      0      ; Drive-Code
FE6D  FF          DRVAT:  DB      neu    ; letztes Laufwerk
FE6E              DRVTAB: rept    16     ; 16 Laufwerke
FE6E                      db      neu
FE6E                      endm
FE6E  FF        A         db      neu
FE6F  FF        A         db      neu
FE70  FF        A         db      neu
FE71  FF        A         db      neu
FE72  FF        A         db      neu
FE73  FF        A         db      neu
FE74  FF        A         db      neu
FE75  FF        A         db      neu
FE76  FF        A         db      neu
FE77  FF        A         db      neu
FE78  FF        A         db      neu
FE79  FF        A         db      neu
FE7A  FF        A         db      neu
FE7B  FF        A         db      neu
FE7C  FF        A         db      neu
FE7D  FF        A         db      neu
FE7E  0000        spsav:  dw      0      ; hier wird Stack gerettet
FE80    0040      xbuff:  ds      64     ; Hilfsbuffer und Stack fuer ROM
        FEC0      tstack  equ     $

                  ;***************************************************************
                  ;* Ende des hardware-Treibers                                  *
                  ;***************************************************************
 Error(s) Detected.
65216 Absolute Bytes. 179 Symbols Detected.
```

Cross Reference:

```
FC7F  .AUXIN       56   190  197
FC77  .AUXIST      57   178  196
FC87  .AUXOUT      55   201  209
FCAB  .CASIN       61   262  269
FCA6  .CASIST      60   251  268
FCB3  .CASOUT      62   273  282
FC4E  .CONIN       52   125
FC3F  .CONIST      51   105
FC74  .CONOST      59   159
FC5D  .CONOUT      53   138
FCDC  .FLOPPY      64   416  872  877
FCD3  .INIT        72   341
FC91  .LIST         2    54  220
FCA0  .LSTST       58   233
FCBD  .SETBNK      63   301
FC36  @CBNK        81   534
FC35  @DBNK        80   451
FC0F  AUXIN        56
FC12  AUXIST       57
FC0C  AUXOUT       55
00C8  BBPORT        2   331
0007  BELL          2
0020  BLANK         2
003F  BLEN          2
FE4D  BOOT        859
0008  BS            2
FD80  BUSY        590   592
FD85  BUSY1       593   595
00CA  CASBAS        2     2    2
FC1E  CASIN        61
FC18  CASIST       60
FC21  CASOUT       62
FC6A  CC          117   136  153  230
FE6C  CCODE       640   684  793  846  890
0048  CENDAT        2
0049  CENIN         2
0049  CENSTB        2   239
FCCC  CL1         311   330
FCCE  CL2         321   331
FCC9  CLOSE       154   323
001A  CLS           2
00CA  CMDCAS        2   257  279
0001  CNTRLA        2
0003  CNTRLC        2
0010  CNTRLP        2
FC03  CONIN        52
FC00  CONIST       51
FC18  CONOST       59
FC06  CONOUT       53
000D  CR            2
FC34  CURBNK       79   310  319  329
000C  CURR          2
00CB  DATCAS        2   270  284
FD60  DEL1        548   549
FD5D  DELAY       540   554  689  700
FD77  DOIT        567   580
FE6D  DRVAT       643   685  891
FE6E  DRVTAB      651   677  892
```

Cross Reference:

```
FE41  DSEL          453  585  690  840
0000  ENTRY         879
FE69  ERRCOD        818  824  887
001B  ESC             2
0000  EXTEND          2
E000  EXTSEG          2
00C0  FCBAS           2    2    2    2    2    2    2
00DE  FCI            33  135
0321  FCO            34  147
00BD  FCTST          35  116
00C0  FDCCMD          2  430  437  502  506  520  524  588  590
                    593
00C3  FDCDAT          2  457  565
00C4  FDCIRQ          2
00C2  FDCSEC          2  456
00C4  FDCSEL          2  434  476  503  521  712  848  850
00C1  FDCTRK          2  661  681  748
FE6B  FEHZA         638  835  837  889
FC37  FLG0           83
FC38  FLG1           84
FC39  FLG2           85
FC3A  FLG3           86
FC3B  FLG4           87
FC3C  FLG5           88
FC3D  FLG6           89
FC3E  FLG7           90
0218  FLO            38  229
FCE7  FLOP1         434  438
FCF1  FLOP2         436  440
FC27  FLOPPY         64
00D0  FORCEI          2  429
020E  FPO            37
0205  FRI            36
FD38  GETDATA       490  516
001E  HHOME           2
0008  HOME            2  577
FC30  INIT           72
FC33  IOBYTE         78
0040  IOEC            2    2    2    2
FC00  IOSEG           2
FFFF  JA              2
0068  KEYBAS          2    2    2
0068  KEYD            2
0069  KEYS            2
000A  LF              2
FC09  LIST           54
FC15  LSTST          58
0020  MINIS           2
F400  MONSEG          2
0040  MOTON           2
0000  NEIN            2    2    2    2
FFFF  NEU           407  644  749  891  895  896  897  898  899
                    900  901  902  903  904  905  906  907  908
                    909  910
FDEF  NODUM         714  740
FE08  NOGOOD        782
FDBD  NOTDEF        645  666
0000  NULL            2
```

Cross Reference:

```
FDE9  NURI          702   716   728
FDE3  NURSEEK       648   705
FCC2  OPEN          115   134   146   228   313
0080  PRMBAS          2     2     2     2
0081  PROMA1          2
0082  PROMA2          2
0080  PROMD           2
FD3D  RD1           521   523
FD42  RD2           524   528   531
FD4D  RD3           526   530
FD14  RDFLP         483   804
00C0  READID          2
0088  READS           2   518
00E0  READT           2
FE22  RETERR        806   815
FD75  RSTORE        571   771
FD52  RWDONE        514   533
0010  SDENS           2
F000  SECBUF          2   873
001C  SEEK            2   566
FD6A  SEEKEX        557   726   760
0001  SELA            2
0002  SELB            2
0004  SELC            2
0008  SELD            2
FD0C  SETIUP        467   470
FC24  SETBNK         63   452   535   869
FE39  SETERR        773   821   829
FCF4  SETUP         445   489   498
0080  SIDE2           2
FE0F  SK11          752   762   787   823
0010  SLEN            2
FD98  SOFT1         610   627
FD8C  SOFTEX        432   599
FE7E  SPSAV         113   132   144   156   226   911
FE6A  STEPRATE      586   623   888
0009  TAB             2
0000  TEST            2     2
0100  TPA             2
FDF8  TRY           754   775
FDFF  TRY1          750   764   820
FEC0  TSTACK        114   133   145   227   913
00FC  UBAS            2     2
00EC  UBAS1           2     2
00CC  UBAS2           2     2     2     2     2
00FE  UCMD            2
00EE  UCMD1           2   351
00CE  UCMD2           2
00FF  UCON            2
00EF  UCON1           2   349
00CF  UCON2           2
00FC  UDAT            2     2     2     2
00EC  UDAT1           2     2     2     2   198   211
00CC  UDAT2           2
000B  UP              2
00FD  USTA            2
00ED  USTA1           2   184   207
00CD  USTA2           2
```

Cross Reference:

```
FD21  WR1        503  505
FD26  WR2        506  510  513
FD31  WR3        508  512
FD19  WRFLP      492  814
00A8  WRITES       2  500
00F0  WRITET       2
FE1F  WRTEX      798  808
FE80  XBUFF      912
```

Kapitel 2 ... Formatieren zum Beispiel

In (fast) allen Kapiteln dieses Buches ist von Tracks und Sektoren die Rede.

Was das eigentlich ist, steht im Glossar, wie diese Tracks und Sektoren aber auf die Diskette kommen, davon war bisher noch nicht die Rede.

Noch einmal kurz zur Klarstellung:

Disketten werden über den sog. Schreib- Lesekopf beschrieben bzw. gelesen. So ein Kopf kann nur in ganz bestimmten, Laufwerksabhängigen Abständen platziert werden. Üblicherweise sind die Abstände so eingerichtet, daß je nach Laufwerk 40, 77 oder 80 Positionen eingestellt werden können.

Unter jeder Kopfposition kann man sich nun eine Spur vorstellen, die, anders wie bei der Schallplatte, einen in sich geschlossenen 'Ring' bildet. Unter diesem Ring (oder auch Zylinder) versteht man nun den sog. Track (engl: Spur,Gleis).

Auf einen Track würden je nach Diskettentyp zwischen 6 und 10 K Dateninformation passen, wenn man diese Datenmenge irgendwie auf die Diskette bringen könnte. Die handelsüblichen Kontroller-Bausteine sind nun jedoch nur in der Lage Datenmengen von 128 bis 1024 Bytes auf einmal zu schreiben oder zu lesen.

Aus diesem Grunde ist jeder Track in mehrere Sektoren aufgeteilt, natürlich auch aus dem Grund die Datenmengen in kleinere Einheiten zu unterteilen.

Diese Sektoren sind nun natürlich nicht von selbst auf die Diskette gekommen, sie werden vielmehr erst mit einem speziellen Programm 'aufgeschrieben'. Dieser Vorgang heißt Formatieren.

Um die Sektoren wieder zu finden, ist auf jeder Diskette ein durchgehendes Loch, das sog. Indexloch, angebracht. Wird dieses Loch erkannt, weiss der Kontroller, daß nun sofort der erste Sektor auf einem Track beginnt.

Ab dieser Stelle werden vom Formatierer, (dem Spezialprogramm) auf magnetischem Wege soviele Sektoren auf die Diskette geschrieben, wie dies von (der speziellen BIOS-Anpassung des) CP/M benötigt wird.

Damit der Kontrollerbaustein die einzelnen Sektoren auseinander halten kann, ist ein ganz bestimmtes Format, wie dies zu geschehen hat, vorgeschrieben. Viele GAP's, Leerzeichen und Kennzeichen müssen mit einem ganz bestimmten Befehl an den Kontrollerbaustein geschrieben werden.

Was da auf die Diskette geschrieben wird, soll der nachfolgende Dump zeigen:

```
0000: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0010: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0020: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0030: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0040: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0050: 00 00 00 00 00 00 00 00 00 00 00 00 F6 F6 F6 FC  ................
0060: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0070: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0080: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0090: 4E 4E 00 00 00 00 00 00 00 00 00 00 00 00 F5 F5  NN..............
00A0: F5 FE 27 00 01 03 F7 4E 4E 4E 4E 4E 4E 4E 4E 4E  ..'....NNNNNNNNN
00B0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 00 00 00  NNNNNNNNNNNNN...
00C0: 00 00 00 00 00 00 00 00 00 F5 F5 F5 FB E5 E5 E5  ................
00D0: E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  ................

      es wird 1024 mal 'E5' geschrieben

04B0: E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  ................
04C0: E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 F7 4E 4E  ..............NN
04D0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
04E0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
04F0: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E  NNNNNNNNNNNNNNNN
0500: 4E 4E 4E 4E 00 00 00 00 00 00 00 00 00 00 00 00  NNNN............
0510: F5 F5 F5 FE 27 00 02 03 F7 4E 4E 4E 4E 4E 4E 4E  ....'....NNNNNNN
0520: 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 00  NNNNNNNNNNNNNNN.
0530: 00 00 00 00 00 00 00 00 00 00 00 F5 F5 F5 FB E5  ................
0540: E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  ................

      dies ist bereits der naechste Sektor
```

Dies ist die Formatierung für einen Sektor auf Seite 0 einer Diskette im NDR-Format.

Im Einzelnen läßt sich der Inhalt wie folgt beschreiben:

80	Bytes	4E	Start-Lücke nach Indexloch
12	Bytes	00	Preambel
3	Bytes	F6	Track Kenner
1	Byte	FC	Index-Kennung
50	Bytes	4E	Warten auf Sektor

Das folgende Feld wiederholt sich für jeden Sektor

12	Bytes	00	
3	Bytes	F5	Sektorkenner
1	Bytes	FE	Start ID-Feld
1	Byte	00..	Track Nummer
1	Byte	00(01)	Seitennummer 0 oder 1
1	Byte	01..	Sektornummer
1	Byte	F7	Prüfsumme (2 Bytes geschrieben)
22	Bytes	4E	Warten
12	Bytes	00	
3	Bytes	F5	Datenkenner
1	Byte	FB	Beginn des Datenfeldes
1024	Bytes	E5	Datenfeld
1	Byte	F7	Prüfsumme Datenfeld (2 Bytes)
54	Bytes	4E	Warten

Ende eines Sektors

Rest der Diskette auffüllen mit 4E's (200..800 mal)

Wichtig in diesem Zusammenhang ist die Tatsache, daß jeder Sektor einen ganz bestimmten Kopf hat, das sog. ID-Feld. Es kann mit einem speziellen Kontollerbefehl gelesen werden und gibt Information über:

den Track, über welchem sich der Kopf befindet

die Seitennummer (0 oder 1)

die Sektornummer

Angaben zur Sektorläne (00=128Bytes, 01=256Bytes usw)

und dann noch eine Prüfsumme anhand derer der Kontroller feststellt, ob die Daten im ID-Feld den auch richtig sind.

Der eigentliche Sektor (also die 128..1024 Bytes) werden üblicherweise mit dem Zeichen E5H gefüllt.

Ein Programm, das diese Arbeit vornimmt wird nachfolgend abgedruckt.

Der hardware-unabhängige Teil wird (aus Copyright-Gründen) nur als DUMP einer REL-Datei abgedruckt, die hardware-Anpassung als Assemblerdatei. Zusammengebunden werden die Programmsegmente mit dem Linker LINK.COM, wobei die Datei FORMAT.REL als erste eingebunden werden muß.

```
0100: 85 91 93 D4 93 50 55 20 24 34 98 09 0D 3E 02 53  .....PU $4...>.S
0110: 4D 9A D9 03 8C 6D 90 3B 36 9F 02 08 6B 38 23 36  M....m.;6...k8#6
0120: A5 02 08 40 00 06 6D 4A 04 03 02 99 B4 4C 10 42  ...@..mJ.....L.B
0130: 00 00 33 6A 50 20 8D 9B 4C 81 1F C5 A1 00 2C FC  ..3jP ..L.......,.
0140: B7 14 3A 40 C0 A6 6D 13 04 0C 3A 9F C5 71 43 79  ..:@..m...:..qCy
0150: FC 58 65 00 00 0D 62 08 E1 C2 21 00 00 17 C3 80  .Xe...b...!.....
0160: 40 86 80 02 B3 6A 50 20 61 A6 32 A9 60 ED D1 08  @....jP a.2.`...
0170: 00 00 26 00 1B C5 22 92 E9 50 52 19 75 B7 43 20  ..&...."..PR.u.C

0180: 60 53 36 89 82 08 69 58 2B 36 A5 02 37 4D C0 06  `S6...iX+6..7M..
0190: E9 98 02 7E 7A B3 6A D0 23 C4 46 FE 05 08 1E A2  ...~z.j.#.F.....
01A0: 1A 82 0A CD A9 40 8C DA B0 06 28 03 BF 89 C2 07  .....@....(.....
01B0: BB 0E 89 00 08 69 F0 33 36 A5 02 01 80 2C DA B0  .....i.36....,..
01C0: 06 28 02 BF 89 C2 07 B8 14 F0 32 A9 80 E2 1A B6  .(........2.....
01D0: 0A CD A9 40 8C DA 64 08 FE 26 B2 A8 90 03 58 82  ...@..d..&....X.
01E0: 38 7A 3F 82 23 80 40 86 C8 82 B3 6A 50 20 61 D0  8z?.#.@....jP a.
01F0: 32 A9 A0 E2 10 00 00 9E 06 00 02 41 20 93 F2 DC  2..........A ...

0200: 50 E8 10 DC 80 56 6D 4A 04 1D 54 D0 76 31 05 9B  P....VmJ..T.v1..
0210: 56 81 04 37 8C 15 98 52 81 19 B5 60 0C 50 07 7F  V..7...R...`.P..
0220: 13 99 54 48 00 30 F4 10 D0 60 66 6D 4A 04 66 D3  ..TH.0...`fmJ.f.
0230: 20 46 6D 3F 01 1D 54 60 75 B8 86 E8 03 08 03 42   Fm?..T`u......B
0240: 1B 42 0C CD A9 40 8C DA B0 06 28 04 BF 89 C2 07  .B...@....(.....
0250: BB 36 A3 82 06 00 6C DB BA 04 21 BC E0 CC DA 94  .6....l...!.....
0260: 08 CD A6 40 8F E2 C0 A0 20 FE 26 B2 A8 00 03 F8  ...@.... .&.....
0270: AE 20 78 01 81 8C DA 26 08 18 4B B3 40 00 06 18  . x....&..K.@...

0280: 00 00 AF 19 54 A0 71 95 49 07 19 54 60 70 30 31  ....T.q.I..T`p01
0290: 9B 44 C1 04 35 D4 19 98 52 81 07 55 28 1D 8C 30  .D..5...R..U(..0
02A0: 66 D5 A0 46 6D 47 04 66 D6 10 36 E9 78 10 DD 2B  f..FmG.f..6.x..+
02B0: 02 47 55 28 1D 6E 28 03 37 4B C0 A6 E9 58 16 DD  .GU(.n(.7K...X..
02C0: 3F 00 D6 E2 00 2B 36 A8 01 06 00 6C DB 5C 02 CD  ?....+6....l.\..
02D0: B9 80 63 AA 92 0E 3C 19 54 90 76 EA F8 08 38 0C  ..c...<.T.v...8.
02E0: 0E AA 50 3A DD 80 3A A9 80 EB 76 43 74 FC 05 1E  ..P:..:...vCt...
02F0: 30 06 55 24 1C 78 32 A9 40 E1 85 2B 34 00 00 0C  0.U$.x2.@..+4...

0300: 2D 1B A4 60 33 64 42 00 00 04 64 80 20 16 CC 04  -..`3dB...d. ...
0310: 02 50 00 0D 89 CE D5 80 04 18 00 1B 00 1D AB 00  .P..............
0320: 08 0C 00 36 7B 3B 56 03 67 E0 8C 26 01 19 00 06  ...6{;V.g..&....
0330: C4 E7 6A C0 02 0C 00 0D 80 0E D5 80 04 06 00 1B  ..j.............
0340: 3D 5D AB 01 B3 F8 46 13 1D 54 90 73 B8 8C 26 3A  =]....F..T.s..&:
0350: A9 40 E7 71 18 4D B2 1A 09 B6 4E E2 30 9B 74 FC  .@.q.M....N.0.t.
0360: 07 3B 88 C2 63 67 B8 8C 26 01 0B 00 06 C4 E7 6A  .;..cg..&......j
0370: C0 02 0C 00 0D 80 0E D5 80 04 06 00 1B 3D 5D AB  .............=].

0380: 01 B3 EC 46 13 6C B7 4F C0 71 0A 00 00 3C 1E 8A  ...F.l.O.q...<..
0390: 00 62 90 C3 E9 CA D9 60 8D 9C AE D5 80 D9 EE 23  .b.....`.......#
03A0: 09 80 46 C0 01 B1 39 DA B0 6C 84 01 4D 90 0B A0  ..F...9..l..M...
03B0: 06 36 27 3B 56 0D 96 4B 64 30 88 6E 91 80 CD 91  .6';V..Kd0.n....
03C0: 0D B6 01 08 C9 00 40 2A A8 10 04 50 00 1B 3F DD  ......@*...P..?.
03D0: AB 00 08 18 00 36 00 3B 56 03 67 E0 8C 26 01 0D  .....6.;V.g..&..
03E0: 00 06 CF F7 6A C0 02 06 00 0D 80 0E D5 80 D9 FC  ....j...........
03F0: 23 09 8E AA 48 39 DC 46 13 1D 54 A0 73 B8 8C 26  #...H9.F..T.s..&

0400: D9 0D 04 DB 27 71 18 4D BA 7E 03 9D C4 61 31 B3  ....'q.M.~...a1.
0410: DC 46 13 00 82 C0 03 67 FB B5 60 01 03 00 06 C0  .F.....g..`.....
0420: 07 6A C0 6C FB 11 84 DB 2D D3 F0 1C 42 80 00 0F  .j.l....-...B...
0430: 07 A2 80 18 A4 30 FA 72 B6 58 23 67 2B B5 60 36  .....0.r.X#g+.`6
0440: 7B 88 C2 60 10 D8 00 6C FF 76 AC 1B 21 00 53 64  {..`...l.v..!.Sd
0450: 02 E8 01 8D 9F EE D5 83 65 92 D9 0C 25 80 C0 C6  ........e...%...
0460: 6D 13 04 57 8C AA 48 38 CA A5 03 8C AA 30 38 CA  m..W..H8.....08.
0470: A3 83 88 68 B8 3B 36 A5 02 0E AA 50 3B 18 60 CD  ...h.;6....P;.`.
```

```
0480: AB 40 8C DA 8E 08 CD AC 20 6D D2 F0 21 BA 56 04  .@...... m..!.V.
0490: 8E AA 50 3A DC 50 06 6E 97 81 4D D2 B0 2D BA 46  ..P:.P.n..M..-.F
04A0: 03 33 6F 30 18 EA A4 83 8F 06 55 24 1D BA BE 02  .3o0......U$....
04B0: 0E 03 63 AA 94 0E B7 10 06 87 55 30 1D 6E 28 0A  ..c.......U0.n(.
04C0: 37 4F C0 51 E0 80 1C 32 A9 20 E3 C1 95 4A 07 0C  7O.Q...2. ...J..
04D0: 2B 59 B6 A8 04 30 BC 66 D4 70 41 0D 2B 07 1D 54  +Y...0.f.pA.+..T
04E0: 60 75 BB 20 43 74 01 98 75 28 11 9B 4C 81 1F C4  `u. Ct..u(..L...
04F0: A6 43 F8 B2 C9 1D 54 D0 72 78 18 00 21 B8 80 00  .C....T.rx..!...

0500: 90 48 24 FC 3C 1E B2 04 64 61 19 38 74 00 00 37  .H$.<...da.8t..7
0510: 46 80 26 E8 D4 04 20 01 8E 80 00 05 81 1C 74 00  F.&... .......t.
0520: 00 37 46 80 36 E8 D4 06 20 01 8E 80 00 05 81 1C  .7F.6... .......
0530: 75 52 81 D6 E2 80 18 E8 00 00 54 11 DD A4 3A 88  uR........T...:.
0540: 0E CD 00 00 04 00 46 68 00 00 C9 10 D8 A0 66 6D  ......Fh......fm
0550: 4A 04 10 DE 70 66 6D 4A 04 66 D3 20 47 F1 61 95  J...pfmJ.f. G.a.
0560: 4E 40 5F C4 D6 55 00 00 7F 15 C5 09 70 C3 B5 B2  N@_..U......p...
0570: ED 25 D4 40 76 6D A5 03 10 D5 80 26 68 00 00 28  .%.@vm.....&h..(

0580: 01 8C AA 30 38 F8 54 CD AB 40 8D 96 4B 65 DA 4B  ...08.T..@..ke.K
0590: A8 80 EC DB A4 06 D9 0D 36 44 37 54 0D 9A 00 00  ........6D7T....
05A0: 0A 00 63 2A 8E 0E D9 09 84 10 E2 1A 8E 0E 7E 1B  ..c*..........~.
05B0: 00 16 E3 E3 B0 A0 0A 32 A8 C0 EE 62 7B 0E AD 02  .......2...b{...
05C0: 39 58 AC DA 48 08 21 00 00 19 B5 28 11 82 CD A4  9X..H.!....(....
05D0: 80 8E 16 48 84 00 00 66 D4 A0 41 F0 29 9B 56 81  ...H...f..A.).V.
05E0: 02 0F 96 4B 36 AC 02 3F 8C 2D 87 F1 ED A0 E6 2F  ...K6..?.-...../
05F0: B2 5C A2 10 00 01 9B 52 81 1C 2C 91 0D 88 06 3F  .\.....R..,....?

0600: 39 8F EC 87 AB 36 AD 02 3C 57 CC 01 18 61 E4 3E  9....6..<W...a.>
0610: 7E B9 5A AC 52 F8 38 0C CD 02 80 18 2D 17 0B 24  ~.Z.R.8.....-..$
0620: 1A 0A 05 08 04 02 01 00 80 40 20 10 08 04 02 A1  .........@ .....
0630: 50 A8 54 20 2A 9B 8D 27 63 29 C8 E6 61 36 19 8D  P.T *..'c)..a6..
0640: E7 23 69 84 E8 69 32 9C 8C A7 21 01 58 CA 72 39  .#i..i2...!.X.r9
0650: 9A 4D E6 E1 00 C4 5C 31 10 0A 85 42 A1 50 34 14  .M....\1...B.P4.
0660: 20 10 08 04 02 01 00 80 40 20 10 08 04 02 01 00   .......@ ......
0670: 80 40 20 10 19 8E A6 53 90 80 AE 44 10 0C 46 E3  .@ ....S...D..F.

0680: 92 C0 80 5A 20 25 9B CD C7 43 91 BC D8 6C 32 9C  ...Z %...C...12.
0690: 81 A0 A0 50 80 40 20 10 08 04 02 01 00 80 40 20  ...P.@ .......@
06A0: 10 08 04 02 82 18 A4 40 27 1C 0D 44 05 23 09 EC  .......@'..D.#..
06B0: EA 6C 10 13 C5 C2 02 59 BC CA 72 31 19 4E 42 C1  .l.....Y..r1.NB.
06C0: 01 10 CA 74 36 9B CD 86 40 68 29 14 07 06 82 88  ...t6...@h).....
06D0: C6 F3 91 B4 C2 74 10 1B 8D 26 33 41 D0 40 64 32  .....t...&3A.@d2
06E0: 99 8D 26 E3 49 94 E4 74 10 0B 44 06 23 49 D0 E8  ..&.I..t..D.#I..
06F0: 65 10 1B 8D E6 33 41 B4 C2 6C 1D 28 0C E6 53 B9  e....3A..l.(..S.

0700: 84 CA 68 36 10 04 07 73 A9 C8 C8 65 10 11 8D E7  ..h6...s...e....
0710: 23 69 84 E8 A0 06 82 8D 27 33 A0 80 C8 61 39 88  #i......'3...a9.
0720: 0E 46 93 19 A0 E8 69 33 88 07 E2 01 41 28 5E 4E  .F....i3....A(^N
0730: 14 A8 01 A6 93 70 80 EE 65 36 18 CD 06 53 68 80  .....p..e6...Sh.
0740: 98 61 3A 99 8E E6 53 91 AC 40 66 37 9C 8D A6 13  .a:...S..@f7....
0750: A1 A4 CA 72 32 9B 84 03 F1 00 A0 9A 29 10 13 4C  ...r2.......)..L
0760: A6 E3 A9 94 40 20 10 08 04 02 01 00 80 40 A0 03  ....@ .......@..
0770: 83 49 86 13 A9 98 EE 65 39 1A C4 06 E3 49 8C D0  .I.....e9....I..

0780: 74 10 1A 4D C7 33 A1 84 D8 6C 34 99 4E 47 41 60  t..M.3...l4.NGA`
0790: 80 C4 69 3A 1D 0C A2 03 71 BC C6 68 36 98 4D 82  ..i:....q..h6.M.
07A0: E1 70 80 50 4D 14 88 09 A6 53 71 D5 CA 0D 29 9B  .p.PM....Sq...).
07B0: CD 86 C1 01 10 D2 73 35 99 4E 87 43 28 80 D2 6E  ......s5.N.C(..n
07C0: 10 13 0C 27 53 31 DC CA 72 35 A8 04 07 73 49 C8  ...'S1..r5...sI.
07D0: D6 6C 34 98 CD 02 03 31 BC E4 6D 30 9D 0D 26 53  .l4....1..m0..&S
07E0: 91 D0 40 77 32 9C 8C 86 53 70 FC 40 28 25 0B C9  ..@w2...Sp.@(%..
07F0: C2 95 00 34 14 42 34 9D 0E 86 51 01 10 D2 73 35  ...4.B4...Q...s5
```

```
0800: 99 4E 87 43 28 80 CA 69 37 1B 0C A6 73 29 B8 40  .N.C(..i7...s).@
0810: 2D 10 18 8C A7 33 A1 84 CA 74 34 99 CC A6 E1 01  -....3...t4.....
0820: B4 02 74 10 1A 8C A6 43 29 C8 40 54 30 9C CE 86  ..t....C).@T0...
0830: 55 00 34 A6 6F 36 1B 04 04 43 49 CC D6 65 3A 1D  U.4.o6...CI..e:.
0840: 0C A2 01 90 B4 E6 65 34 9D 0D 26 71 01 98 DE 72  ......e4..&q...r
0850: 36 98 4E 86 93 29 C8 E8 20 3B 99 4E 46 43 29 B8  6.N..).. ;.NFC).
0860: 40 3F 10 0A 09 42 F2 70 A4 40 20 10 08 04 0A 00  @?...B.p.@ .....
0870: 68 28 8C 6F 39 1B 4C 27 43 49 94 E4 65 10 14 CC  n(.o9.L'CI..e...

0880: A6 93 A1 95 40 0D 45 01 C1 A0 A2 61 84 EA 66 3B  ....@.E....a..f;
0890: 99 4E 46 B3 31 94 D0 6C 32 9C 84 02 01 00 80 40  .NF.1..l2......@
08A0: A0 06 82 8E 66 F3 61 B0 40 46 37 9C 8D A6 13 A1  ....f.a.@F7.....
08B0: A4 CA 72 3A 98 8C E2 03 A9 94 C4 65 39 1C 0E 47  ..r:.......e9..G
08C0: 53 29 98 E8 20 3B 99 4E 46 43 29 B8 40 3F 10 0A  S).. ;.NFC).@?..
08D0: 09 42 F2 70 A5 40 07 06 82 88 C6 F3 91 B4 C2 74  .B.p.@.........t
08E0: 34 99 4E 47 53 71 9C E6 66 32 9A 0D 86 53 90 80  4.NGSq..f2...S..
08F0: 50 57 14 88 0E E6 93 29 90 CA 72 34 1B CD 86 53  PW.....)..r4...S

0900: 70 80 50 4D 14 88 09 A6 53 71 D4 CA 20 14 16 05  p.PM....Sq.. ...
0910: 22 03 29 E0 D2 74 10 08 04 02 01 00 80 40 20 10  ".)..t.......@ .
0920: 08 04 02 01 02 80 1A 0A 2A 99 4C 46 53 91 C0 E4  ........*.LFS...
0930: 75 32 99 8C A2 02 99 94 D2 74 32 A8 08 C6 F3 91  u2.......t2.....
0940: B4 C2 74 34 99 4E 47 53 71 9C 40 65 34 9B 8E E6  ..t4.NGSq.@e4...
0950: 13 71 90 CC 72 32 9A 54 00 00 00 00 00 00 00 00  .q..r2.T........
0960: 00 00 00 04 64 94 12 A1 A4 27 A6 A2 C7 56 01 12  ....d....'...V..
0970: 1A 4C 64 64 12 A1 A6 22 A7 A9 C6 50 01 22 1A 62  .Ldd..."...P.".b

0980: 92 9C 75 68 11 21 A7 C6 61 80 EA 22 22 2A 72 9C  ..uh.!..a..""*r.
0990: 65 0C 0F 22 27 A9 AA 22 A8 46 5A 00 F2 22 92 B2  e.."'..".FZ.."..
09A0: A2 0A 14 67 B0 0F 22 29 A5 A9 A2 A6 46 46 00 2A  ...g..")....FF.*
09B0: 32 12 6A 9A 3C 65 40 03 23 29 26 AA 20 A1 46 7C  2.j.<e@.#)&. .F|
09C0: 80 EA 4A 6A 0A 3A 2C 64 F8 03 26 A0 AC 23 29 26  ..Jj.:,d..&..#)&
09D0: C6 60 40 E2 6A 0A C2 4C 65 D8 0E 26 A4 A7 24 C6  .`@.j..Le..&..$.
09E0: 4E 80 72 6A 7A A2 7A 32 34 67 D4 0E A9 22 29 A2  N.rjz.z24g...").
09F0: A1 C6 6A 80 F2 92 2A 9A A2 7A 94 66 44 0E A9 A2  ..j...*..z.fD...

0A00: 22 A7 29 C6 67 00 F2 9A 4A 22 6A 9A 5C 75 28 11  ".).g...J"j.\u(.
0A10: 29 A6 C6 43 40 32 AA 9A 92 62 4A 74 67 60 0F 2B  )..C@2...bJtg`.+
0A20: A9 2A 2A 29 25 CE 00 00 00 9E 00 00 00 00 00 00  .**)%...........
0A30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0A40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0A50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0A60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0A70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
                  ; ************************************************************
                  ; *                                                          *
                  ; * Format- und hardware-Anpassung an den Universal-Formatierer *
                  ; *                                                          *
                  ; * Es duerfen alle PRIME-Register verwendet werden          *
                  ; * KEINESFALL den 2. Registersatz und die Register IX und IY *
                  ; * verwenden. Es stehen 16*2 Bytes Stacktief zur Verfuegung  *
                  ; *                                                          *
                  ; ************************************************************
                  ;


                          ;
                          ; ***************
                          ; Vereinbarungen
                          ; ***************
                          ;

       FFFF       ja      equ     -1
       0000       nein    equ     not ja
       000D       cr      equ     0dh
       000A       lf      equ     0ah
       0007       bell    equ     07h


                          ;
                          ; ***************
                          ; Linkadressen
                          ; ***************
                          ;



                          ;
                          ; Auf folgende Unterprogramme oder Labels greift das Haupt-
                          ; programm zurueck. Sie muessen den angegebenen Spezifikationen
                          ; entsprechen.
                          ;

                          public  sidmsk,mini,maxi,sdens,ddens
                          public  drvtab,frmtab,maxfrm,fbmsg,usrlin
                          public  conin,clrs,chome,cleos
                          public  dsksel,dostep,restor,wrttrk,rdsec,motoff
                          public  image

                          ;
                          ; ***************
                          ; Vereinbarungen
                          ; ***************
                          ;


                          ;
                          ; folgende Daten muessen sytemabhaengig angepasst werden
                          ;


                          ;
                          ; **** Laufwerke
                          ;

       0001       sela    equ     00000001b       ; Laufwerksselekt 'A' Seite 0(1)
```

```
          0002      selb    equ     00000010b       ; Laufwerk 'B' - Seite 0(1)
          0004      selc    equ     00000100b       ; Laufwerk 'C' - Seite 0(1)
          0008      seld    equ     00001000b       ; Laufwerk 'D' - Seite 0(1)

                            ;
                            ; auf folgende Werte wird aus dem Hauptprogramm
                            ; zugegriffen, sie muessen daher als Byte vorliegen
                            ;

 0000' 80           sidmsk: db      10000000b

 0001' 20           mini:   db      00100000b       ; Select fuer MINI-Laufwerke
 0002' 00           maxi:   db      00000000b       ; Select fuer MAXI-Laufwerke
 0003' 10           sdens:  db      00010000b       ; Select fuer single-density
 0004' 00           ddens:  db      00000000b       ; Select fuer double-density

                            ;
                            ; ****************
                            ; Laufwerkstabelle
                            ; ****************
                            ;

 0005'              drvtab:

                            ;
                            ; Laufwerke sind spezifiziert wie folgt:
                            ;
                            ; xdrive + 0   JA oder NEIN je nach Vorhandensein
                            ; xdrive + 1   reiner Laufwerksselect ohne side- oder density
                            ;              und ohne MOTOR-ON oder MINI und MAXI Select
                            ; xdrive + 2   steppingrate ( 00000000b..00000011b )
                            ;

 0005' FF           adrive: db      ja
 0006' 01                   db      sela
 0007' 01                   db      00000001b       ; 3ms

 0008' FF           bdrive: db      ja
 0009' 02                   db      selb
 000A' 01                   db      00000001b

 000B' 00           cdrive: db      nein
 000C' 04                   db      selc
 000D' 00                   db      00000000b

 000E' 00           ddrive: db      nein
 000F' 08                   db      seld
 0010' 00                   db      00000000b

 0011' 00           edrive: db      nein
 0012' 00                   db      0
 0013' 00                   db      0

 0014' 00           fdrive: db      nein
 0015' 00                   db      0
 0016' 00                   db      0

 0017' 00           gdrive: db      nein
```

```
0018' 00                db      0
0019' 00                db      0

001A' 00        hdrive: db      nein
001B' 00                db      0
001C' 00                db      0

001D' 00        idrive: db      nein
001E' 00                db      0
001F' 00                db      0

0020' 00        jdrive: db      nein
0021' 00                db      0
0022' 00                db      0

0023' 00        kdrive: db      nein
0024' 00                db      0
0025' 00                db      0

0026' 00        ldrive: db      nein
0027' 00                db      0
0028' 00                db      0

0029' 00        mdrive: db      nein
002A' 00                db      0
002B' 00                db      0

002C' 00        ndrive: db      nein
002D' 00                db      0
002E' 00                db      0

002F' 00        odrive: db      nein
0030' 00                db      0
0031' 00                db      0

0032' 00        pdrive: db      nein
0033' 00                db      0
0034' 00                db      0

                        ;
                        ; ****************
                        ; Formatangaben
                        ; ****************
                        ;

                        ;
                        ; diese Angaben muessen Formatabhaengig angepasst werden
                        ;

                        ; ****************
                        ; Vereinbarungen
                        ; ****************

        0000    f128    equ     0               ; 128 Byte Sektorgroesse
        0001    f256    equ     1               ; 256 Byte Sektorgroesse
        0002    f512    equ     2               ; 512 Byte Sektorgroesse
        0003    f1024   equ     3               ;  1k Byte Sektorgroesse

```

```
0035'                   frmtab:


                                ;
                                ; die folgenden Daten enthalten alle Angaben die zum
                                ; Autbau eines Formatierungimages notwendig sind.
                                ; Die kommentierte Reihenfolge ist einzuhalten
                                ;


0035' 0079'             form1:  dw      fomsga          ; Text der Spezifikation
0037' FF                        db      ja              ; MINI-Laufwerk ?
0038' FF                        db      ja              ; doppelte Dichte ?
0039' 50                        db      80              ; Tracks pro Seite
003A' FF                        db      ja              ; zweiseitig moeglich ?
003B' 05                        db      5               ; Sektoren pro Seite
003C' 03                        db      f1024           ; Sektorlaenge
003D' 005A'                     dw      xltla           ; Sektor SKEW Seite 0
003F' 005A'                     dw      xltla           ; Sektor SKEW Seite 1

0041' 00A0'             form2:  dw      fbmsgb          ; Text der Spezifikation
0043' FF                        db      ja              ; MINI-Laufwerk ?
0044' FF                        db      ja              ; doppelte Dichte ?
0045' 28                        db      40              ; Tracks pro Seite
0046' FF                        db      ja              ; zweiseitig moeglich ?
0047' 05                        db      5               ; Sektoren pro Seite
0048' 03                        db      f1024           ; Sektorlaenge
0049' 005A'                     dw      xltla           ; Sektor SKEW Seite 0
004B' 005A'                     dw      xltla           ; Sektor SKEW Seite 1

004D' 00C7'             form3:  dw      fbmsgc          ; Adresse der Kopfzeile
004F' 00                        db      nein            ; MINI-Laufwerk ?
0050' 00                        db      nein            ; doppelte Dichte ?
0051' 4D                        db      77              ; Tracks pro Seite
0052' 00                        db      nein            ; zweiseitig moeglich ?
0053' 1A                        db      26              ; Sektoren pro Seite
0054' 00                        db      f128            ; Sektorlaenge
0055' 005F'                     dw      xlt2a           ; Sektor SKEW Seite 0
0057' 005F'                     dw      xlt2a           ; ... nur einseitig

          0003          frmlen  equ     ($-frmtab)/12
0059' 03                maxfrm: db      frmlen          ; Endemarke fuer Hauptprogramm

                                ;
                                ; ****************
                                ; Skewtabelle
                                ; ****************
                                ;

005A' 01 02 03 04 xltla:        db      1,2,3,4,5
005F' 01 02 03 04 xlt2a:        db      1,2,3,4,5,6,7,8,9,10
0069' 0B 0C 0D 0E               db      11,12,13,14,15,16,17,18,19,20
0073' 15 16 17 18               db      21,22,23,24,25,26

                                ;
                                ; ****************
                                ; Kopfzeilen
                                ; ****************
                                ;
```

```

                            ;
                            ; Die Texte der Kopfzeilen werden als Menue ausgegeben.
                            ; Der Textaufbau sollte immer der gleichen Anordnung entsprechen
                            ; Pro Format ist eine Zeile vorgesehen die IMMER mit einer
                            ; FORTLAUFENDEN Bezeichnung, beginnend bei 'A' beginnen muss.
                            ; Menues sind bis einschliesslich dem Buchstaben 'Q' zugelassen.
                            ; Es werden pro Menue nur 8 Zeilen ausgegeben. Die neunte Zeile
                            ; muss die den Text:
                            ;
                            ; ' (Z) weiter (W) wiederholen (X) eXit',' '+80h
                            ;
                            ; entnalten und mit dem 7.Bit gesetzt abschliessen. Danach koennen
                            ; weitere 8 Zeilen, ebenfalls mit obigem Abschluss folgen bis
                            ; Menue 'Q'
                            ;
                            ; Nicht vergessen - die letzte Menuezeile mit Bit 7 gesetzt !!!
                            ;

0079'                fbmsg:

0079' 20 28 41 29 fbmsga: db      ' (A)   80 Track 2-seitig Standard NOR',cr,lf
00A0' 20 28 42 29 fbmsgb: db      ' (B)   40 Track 2-seitig Standard NOR',cr,lf
00C7' 20 28 43 29 fbmsgc: db      ' (C)    8" ss-sd Standard CP/M ',cr,lf
00E8' 20 28 44 29 fbmsgd: db      ' (D)',cr,lf
00EE' 20 28 45 29 fbmsge: db      ' (E)',cr,lf
00F4' 20 28 46 29 fbmsgf: db      ' (F)',cr,lf
00FA' 20 28 47 29 fbmsgg: db      ' (G)',cr,lf
0100' 20 28 48 29 fbmsgh: db      ' (H)',cr,lf
0106' 20 28 5A 29         db      ' (Z) weiter (W) wiederholen (X) eXit',' '+80h

012B' 20 28 49 29 fbmsgi: db      ' (I)',cr,lf
0131' 20 28 4A 29 fbmsgj: db      ' (J)',cr,lf
0137' 20 28 4B 29 fbmsgk: db      ' (K)',cr,lf
013D' 20 28 4C 29 fbmsgl: db      ' (L)',cr,lf
0143' 20 28 4D 29 fbmsgm: db      ' (M)',cr,lf
0149' 20 28 4E 29 fbmsgn: db      ' (N)',cr,lf
014F' 20 28 4F 29 fbmsgo: db      ' (O)',cr,lf
0155' 20 28 50 29 fbmsgp: db      ' (P)',cr,lf
015B' 20 28 5A 29         db      ' (Z) weiter (W) wiederholen (X) eXit',' '+80h

0180' 20 28 51 29 fbmsgq: db      ' (Q)',cr,lf
0186' 20 28 52 29 fbmsgr: db      ' (R)',cr,lf
018C' 20 28 53 29 fbmsgs: db      ' (S)',cr,lf
0192' 20 28 54 29 fbmsgt: db      ' (T)',cr,lf
0198' 20 28 55 29 fbmsgu: db      ' (U)',cr,lf
019E' 20 28 56 29 fbmsgv: db      ' (V)',cr,lf
01A4' 20 28 59 29 fbmsgy: db      ' (Y)',cr,lf
01AA' 0D 0A               db      cr,lf
01AC' 20 28 5A 29         db      ' (Z) weiter (W) wiederholen (X) eXit',' '+80h

01D1' 00                  db      0              ; Ende-Marke

                            ;
                            ; Bildschirm loeschen
                            ;

01D2' 0C 9A        clrs:  db      0ch,1ah+80h    ; Abschliessen mit Bit 7 gesetzt
```

```
                              ;
                              ; Cursor ohne loeschen nach Bildschirm links oben
                              ;

0104' 9E          chome:  db      1eh+80h

                              ;
                              ; Bildschirm ab Cursorposition loeschen
                              ; wenn Funktion nicht vorhanden CLRS-Sequenz verwenden
                              ;

0105' 1B D9       cleos:  db      1bh,'Y'+80h

                              ;
                              ; Benutzerzeile - wird nach dem Programmheader ausgegeben
                              ; Muss mit CR,LF enden, mit 7. Bit gesetzt
                              ; Wird die Zeile nicht benutzt muss sie mit einer 0
                              ; abgeschlossen werden
                              ; Der Text darf maximal 2 Zeilen umfassen.
                              ;

0107' 20 20 20 20 usrlin: db      '           mit Anpassung an den NDR-Klein-Computer'
0209' 0D 0A 8A            db      cr,lf,lf+80h    ; (2* linefeed)
```

```
                        ; ******************************************************************
                        ; * hardware-Anpassung an den Universalformatierer                 *
                        ; ******************************************************************

                                ;
                                ; ****************
                                ; Vereinbarungen
                                ; ****************
                                ;

           00C0         fdccmd  equ     0c0h
           00C1         fdctrk  equ     0c1h
           00C2         fdcsec  equ     0c2h
           00C3         fdcdat  equ     0c3h
           00C4         fdcsel  equ     0c4h

           000C         home    equ     00001100b       ; ohne step aber headload & verify
           001C         seek    equ     00011100b
           008C         reads   equ     10001100b
           00C4         readid  equ     11000100b
           0058         stepin  equ     01011000b
           00F4         writetr equ     11110100b

                                ;
                                ; ****************
                                ; CONIN
                                ; ****************
                                ;

 020C' C3 FC03          conin:  jp      0fc03h          ; Bewusst MONIO da nur fuer FDC2 angepasst!

                                ;
                                ; ****************
                                ; Kontroller
                                ; ****************
                                ;

 020F'                  dsksel:

                                ;
                                ; Laufwerk ansprechen
                                ; Erhaelt vom Hauptprogramm Hauptprogramm in Register <B>
                                ; den SELECT - SIDESELECT - DISK (mini-maxi) und DENSITY nach
                                ; den Vorgabewerten in diesem Programm. Es kann notwendig sein
                                ; die entsprechenden Daten erst hier anzupassen, wenn z.B.
                                ; der Select auf andere Weise oder ueber verschiedene Ports
                                ; erfolgt.
                                ; Programm muss mit TIME-OUT arbeiten. Falls Laufwerk nicht
                                ; angesprochen werden kann (keine Diskette oder was immer
                                ; muss in <A>=FF und NZ-Flag zurueckgekehrt werden.
                                ;

 020F' ED 43 029C'              ld      (step),bc       ; Werte retten
 0213' 78                       ld      a,b             ; SELECT
 0214' D3 C4                    out     (fdcsel),a      ; und Laufwerk ansprechen

                                ;
                                ; warten bis READY oder TIME-OUT
```

```
                                ;
                                ; bei festverdrahtetem Ready entsprechende Delayschleife
                                ; (200..500ms) vorsehen, (siehe Datenblatt des Laufwerkes)
                                ;

0216' 21 0000                   ld      hl,0            ; TIME-OUT Schleife
0219' DB C0             dsk1:   in      a,(fdccmd)
021B' E6 80                     and     10000000b       ; Teste READY
021D' C8                        ret     z               ; OK
021E' 3D                dsk2:   dec     a
021F' 20 FD                     jr      nz,dsk2         ; --warteinweilchen--
0221' 2B                        dec     hl
0222' 7C                        ld      a,h
0223' B5                        or      l
0224' 20 F3                     jr      nz,dsk1         ; weiter ...
0226' 3D                        dec     a               ; A=FF und NZ
0227' C9                        ret


0228'                   dostep:

                                ;
                                ; STEP auf naechsten Track
                                ;

0228' 06 58                     ld      b,stepin
022A' 18 05                     jr      busy

022C'                   restor:

                                ;
                                ; Home des Laufwerkskopfes
                                ;

022C' 06 0C                     ld      b,home
022E' CD 0231'                  call    busy

0231'                   busy:

                                ;
                                ; physikalischer Zugriff fuer TYPE I Befehle
                                ;

0231' DB C0                     in      a,(fdccmd)      ; reset moegl. INT
0233' 3A 029C'                  ld      a,(step)        ; stepping-rate
0236' B0                        or      b               ; einfuegen
0237' D3 C0                     out     (fdccmd),a      ; Befehl ausgeben
0239' DB C4             busy1:  in      a,(fdcsel)      ; warten auf INT
023B' CB 77                     bit     6,a
023D' 28 FA                     jr      z,busy1
023F' DB C0                     in      a,(fdccmd)      ; lese Status
0241' E6 90                     and     10010000b       ; Maskiere READY & RNF-SEEK
0243' C9                        ret

0244'                   wrttrk:

                                ;
                                ; Schreiben eines Tracks
```

```
                              ; <HL> zeigt auf Puffer, Track ist eingestellt
                              ;

0244' 06 F4                   ld      b,writetr


0246'               putdata:


                              ;
                              ; physikalischer Schreibzugriff
                              ;

0246' DB C0                   in      a,(fdccmd)      ; Reset moeglichen INT
0248' 3A 029D'                ld      a,(cursel)      ; aus select
024B' E6 80                   and     a,10000000b     ; sideselect extrahieren
024D' 17                      rla                     ; und
024E' 17                      rla                     ; fuer WD 1797
024F' 17                      rla                     ; side select in Bit 1
0250' B0                      or      b               ; in Befehl eintuegen
0251' 0E C3                   ld      c,fdcdat        ; Datenregister
0253' D3 C0                   out     (fdccmd),a      ; ... und los gehts
0255' DB C4         putd1:    in      a,(fdcsel)      ; warten bis Kontroller bereit
0257' 07                      rlca
0258' 30 FB                   jr      nc,putd1
025A' DB C0         putd2:    in      a,(fdccmd)      ; Status lesen
025C' CB 4F                   bit     1,a             ; DRQ?
025E' 28 05                   jr      z,putd3         ; wenn nicht ..
0260' ED A3                   outi                    ; sonst Daten raus
0262' C3 025A'                jp      putd2           ; und weiter geht's

0265' CB 47         putd3:    bit     0,a             ; noch BUSY?
0267' C2 025A'                jp      nz,putd2        ; ... ja, sonst
026A' E6 E4                   and     '1100100b       ; Fehler maskieren
026C' C9                      ret

026D'               rdsec:

                              ;
                              ; lesen eines Sektors zu Testzwecken
                              ; <HL> zeigt auf Puffer
                              ; <A>  =Sektor
                              ;

026D' D3 C2                   out     (fdcsec),a      ; Sektorregister setzen
026F' 0E 8C                   ld      c,reads

0271'               getdata:

                              ;
                              ; physikalisches Lesen
                              ;

0271' 3A 029D'                ld      a,(cursel)      ; sso-Bit fuer
0274' E6 80                   and     10000000b       ; WD 1797 einfuegen
0276' 17                      rla
0277' 17                      rla
0278' 17                      rla
0279' B1                      or      c               ; sso-Bit einfuegen
```

```
027A' 0E C3                ld      c,fdcdat        ; Datenport
027C' D3 C0                out     (fdccmd),a      ; Befehl ausgeben
027E' DB C4       getd1:   in      a,(fdcsel)      ; abwarten bis
0280' 07                   rlca                    ; DRQ ?
0281' 30 FB                jr      nc,getd1        ; wenn noch nicht ..

0283' DB C0       getd2:   in      a,(fdccmd)
0285' CB 4F                bit     1,a             ; DRQ
0287' CA 028F'             jp      z,getd3
028A' ED A2                ini                     ; lesen
028C' C3 0283'             jp      getd2

028F' CB 47       getd3:   bit     0,a
0291' C2 0283'             jp      nz,getd2
0294' E6 9C                and     10011100b       ; maskieren
0296' C9                   ret

0297'             motoff:

                           ;
                           ; Abschalten des Laufwerkmotors
                           ; Falls nicht notwendig mit RET abschliessen
                           ;

0297' 3E 40                ld      a,01000000b
0299' D3 C4                out     (fdcsel),a
029B' C9                   ret


                           ; ***************
                           ; RAM-Bereich
                           ; ***************
                           ;

029C' 00          step:    db      0
029D' 00          cursel:  db      0

029E'             image:

                           ;
                           ; ab diesem Bereich wird das Formatierungsimage aufgebaut
                           ;

                           end
 0 Error(s) Detected. 670 Program Bytes.
 104 Symbols Detected.
```

Cross Reference:

```
0005' ADRIVE      92
0008' BDRIVE      96
0007  BELL        22
0231' BUSY       401  410  412
0239' BUSY1      422  424
000B' CDRIVE     100
01D4' CHOME       40  296
01D5' CLEOS       40  303
01D2' CLRS        40  290
020C' CONIN       40  346
000D  CR          20  255  256  257  258  259  260  261  262
                 265  266  267  268  269  270  271  272  275
                 276  277  278  279  280  281  281  314
029D' CURSEL     446  485  526
0004' DDENS       38   73
000E' DDRIVE     104
0228' DOSTEP      41  394
0005' DRVTAB      39   81
0219' DSK1       381  389
021E' DSK2       384  385
020F' DSKSEL      41  354
0011' EDRIVE     108
0003  F1024      173  190  200
0000  F128       170  210
0001  F256       171
0002  F512       172
0079' FBMSG       39  252
0079' FBMSGA     184  254
00A0' FBMSGB     194  255
00C7' FBMSGC     204  256
00E8' FBMSGD     257
00EE' FBMSGE     258
00F4' FBMSGF     259
00FA' FBMSGG     260
0100' FBMSGH     261
012B' FBMSGI     264
0131' FBMSGJ     265
0137' FBMSGK     266
013D' FBMSGL     267
0143' FBMSGM     268
0149' FBMSGN     269
014F' FBMSGO     270
0155' FBMSGP     271
0180' FBMSGQ     274
0186' FBMSGR     275
018C' FBMSGS     276
0192' FBMSGT     277
0198' FBMSGU     278
019E' FBMSGV     279
01A4' FBMSGY     280
00C0  FDCCMD     327  381  418  421  425  445  453  457  492
                 497
00C3  FDCDAT     330  452  491
00C2  FDCSEC     329  476
00C4  FDCSEL     331  371  422  454  493  516
00C1  FDCTRK     328
0014' FDRIVE     112
0035' FORM1      184
```

```
Z80ASM SuperFast Relocating Macro Assembler 20 Dec 85 09:32 Z80ASM 1.24 Page  12
Cross Reference:

0041' FORM2          194
004D' FORM3          204
0003  FRMLEN         214  215
0035' FRMTAB          39  175  214
0017' GDRIVE         116
027E' GETD1          493  495
0283' GETD2          497  501  504
028F' GETD3          499  503
0271' GETDATA        479
001A' HDRIVE         120
000C  HOME           333  409
001D' IDRIVE         124
029E' IMAGE           42  528
FFFF  JA              18   19   92   96  185  186  188  195  196
                     198
0020' JDRIVE         128
0023' KDRIVE         132
0026' LDRIVE         136
000A  LF              21  255  256  257  258  259  260  261  262
                     265  266  267  268  269  270  271  272  275
                     276  277  278  279  280  281  281  314  314
0059' MAXFRM          39  215
0002' MAXI            38   71
0029' MDRIVE         140
0001' MINI            38   70
0297' MOTOFF          41  508
002C' NDRIVE         144
0000  NEIN            19  100  104  108  112  116  120  124  128
                     132  136  140  144  148  152  205  206  208
002F' ODRIVE         148
0032' PDRIVE         152
0255' PUTD1          454  456
025A' PUTD2          457  461  464
0265' PUTD3          459  463
0246' PUTDATA        439
026D' RDSEC           41  468
00C4  READID         336
008C  READS          335  477
022C' RESTOR          41  403
0003' SDENS           38   72
001C  SEEK           334
0001  SELA            58   93
0002  SELB            59   97
0004  SELC            60  101
0008  SELD            61  105
0000' SIDMSK          38   68
029C' STEP           369  419  525
0058  STEPIN         337  400
0107' USRLIN          39  313
00F4  WRITETR        338  436
0244' WRTTRK          41  429
005A' XLT1A          191  192  201  202  223
005F' XLT2A          211  212  224
```